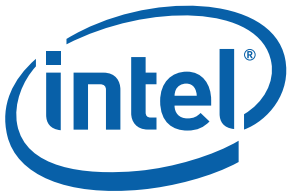


Intel® EP80579 Software for IP Telephony Applications on Intel® QuickAssist Technology for Linux*

Getting Started Guide

March 2009

Order Number: 320414-003US

Contents

1.0 Introduction 7
- 1.1 About this Manual 7
- 1.2 Additional Information on Software 7
  - 1.2.1 Where to Find Current Software and Documentation 7
  - 1.2.2 Product Documentation 7
  - 1.2.3 Pre-boot Firmware 8
- 1.3 Related Software and Documentation 8
  - 1.3.1 Open Source Software and Patches Required 8
  - 1.3.2 Intel® EP80579 Integrated Processor Software Shims 9
- 1.4 Conventions 9
- 1.5 Software Overview 9
  - 1.5.1 Features Implemented 9
  - 1.5.2 List of Files in Release 10
  - 1.5.3 Package Release Structure 10

2.0 Configuration Requirements 12
- 2.1 Development Board Configuration 12
  - 2.1.1 Package Components 12
  - 2.1.2 Development Kit Setup 13
  - 2.1.3 Safety 13
  - 2.1.4 Connecting the Serial ATA Hard Drive and Cable 13
  - 2.1.5 Connecting the Keyboard and Mouse 13
  - 2.1.6 Connecting the PCI Express* Video Card 14
  - 2.1.7 Connecting the Serial ATA DVD-ROM Drive (Optional) 14
  - 2.1.8 Connecting Voice and/or T1/E1 Mezzanine Card(s) 14
  - 2.1.9 Connecting the Power Cables 14
  - 2.1.10 Powering Up the System 14
- 2.2 Development Platform Configuration for IP Telephony Applications 18
- 2.3 Development Board Setup Requirements 19

3.0 Installing the OS on a Development Board 21
- 3.1 System Requirements 21
- 3.2 Acquiring Red Hat Enterprise Linux 5.0 21
- 3.3 Installing Red Hat Enterprise Linux 5.0 21
  - 3.3.1 Recommended Installation Customizations 23
- 3.4 Installing Red Hat Enterprise Linux 5.0 Kernel Source 24
- 3.5 Unpacking the EP80579 Voice Software Linux Package 26
- 3.6 Patching the Kernel for PCI Device Recognition 27
- 3.7 (Optional) Modifications for Use with OCF 28
  - 3.7.1 Applying the OCF Patch 28
  - 3.7.2 Configuring OCF Parameters 28
  - 3.7.3 Backporting OCF Features and Bug Fixes 29
- 3.8 Rebuilding the Kernel 29
  - 3.8.1 Restore the Old Configuration 29
  - 3.8.2 Modifying the Makefile 30
  - 3.8.3 Building and Installing the Patched Kernel Source 30
  - 3.8.4 Making the New Kernel Default 30
  - 3.8.5 Rebooting and Verifying 31

4.0 Building EP80579 IP Telephony Software on a Target Development Board 32
- 4.1 Environment Setup 32
- 4.2 Build Options 32
- 4.3 Build Using Top Level Make 33

4.4 Optionally Installing Accelerated libSrtp 34
4.5 Installation of Build Output 35
4.5.1 IP Telephony 35

5.0 Runtime Configuration 38
5.1 Installing and Configuring Telephony Modules 38
5.2 Loading OCF and OCF Shim (Optional for IP Telephony Applications) 38
5.3 Using the Intel® QuickAssist Technology Cryptographic API 39

6.0 Building Security and Embedded Components Individually 40
6.1 Building Components Individually Using Top-Level Make 40

7.0 Building, Installing and Loading Individual EP80579 Embedded Software Drivers 41
7.1 Controller Area Network (CAN) Driver 41
7.1.1 Linux Compilation Instructions 41
7.1.2 Linux Module Load/Unload Instructions 41
7.2 Enhanced Direct Memory Access (EDMA) Driver 42
7.2.1 Linux Compilation Instructions 42
7.2.2 Linux Module Load/Unload Instructions 42
7.2.3 Runtime Configuration of EDMA 42
7.3 Watchdog Timer (WDT) Driver 43
7.3.1 Linux Compilation Instructions 43
7.3.2 Linux Module Load/Unload Instructions 43
7.3.3 Runtime Configuration of WDT 43
7.4 General Purpose I/O (GPIO) Driver 43
7.4.1 Linux Compilation Instructions 43
7.4.2 Linux Module Load/Unload Instructions 44
7.5 IEEE 1588 Hardware Assist Driver 44
7.5.1 Linux Compilation Instructions 44
7.5.2 Linux Module Load/Unload Instructions 45
7.5.3 Runtime Configuration of IEEE 1588 45
7.6 Global Configuration Unit and Gigabit Ethernet Drivers 45
7.6.1 Linux Compilation Instructions 45
7.6.2 Linux Module Load/Unload Instructions 46
7.6.3 Runtime Configuration of GCU and Gigabit Ethernet 46
7.6.4 Alternative PHY Compilation 46
7.7 System Management Bus (SMBus) Driver 47
7.7.1 Linux Compilation Instructions 47
7.7.2 Linux Module Load/Unload Instructions 48
7.7.3 Runtime Configuration of SMBus 48

8.0 Building and Using Crypto and Debug Tools 49
8.1 Building Crypto Tools 49
8.2 Using the Crypto Tools 49
8.2.1 Using Debug Tools 50

9.0 Pre-boot (BIOS) Firmware 52
9.1 Pre-boot Firmware Setup Menu 52
9.1.1 Serial Console Redirection 53
9.1.2 Changing the Boot Device 53
9.1.3 Maximum Memory Speed Setup 53
9.1.4 Coherent and Non-Coherent Memory Allocation 54
9.1.5 Legacy and AHCI SATA Mode 54
9.2 Pre-boot Firmware Image Reflashing Instructions 55
9.2.1 Aptio Flash Update Utility (AFUEFI) 55

10.0 Uninstalling the Software 57

10.1 Linux Module/Driver Dependencies 57
10.2 Unloading IP Telephony Drivers 57

11.0 Using the HSS Channel Driver for Asterisk 59
11.1 Architectural Information 59
11.2 Design Considerations 61
11.2.1 Functional Model 61
11.2.2 Threading Model 61
11.2.3 Bill of Materials 62
11.3 Debug Considerations 62
11.4 Performance Considerations 62
11.5 Installation Instructions 63
11.5.1 OS, Kernel, Libraries and Tools Release Requirements 63
11.5.2 Enabling Echo Cancellation 63
11.5.3 Building Asterisk Instructions 64
11.5.4 Module Install Requirements 64
11.5.5 Making A Call 65

12.0 Troubleshooting 69
12.1 Using a Graphics Card 69
12.2 Using the Serial Port 69

13.0 Glossary 71

A Build and Install Openswan 72
A.1 Environment Setup 72
A.2 Building and Installing the GMP Library 72
A.3 Building and Installing OpenSSL 73
A.4 Building and Installing Openswan 75
A.4.1 Prepare to Build Openswan 75
A.4.2 Building and Installing Openswan 75

B Configuring Sample VPN Application 77
B.1 Configure IPSec on Both Gateways 77
B.1.1 Testing if IPSec is Configured Correctly 78
B.2 Set Up a Tunnel Between Gateways 79
B.3 Configuring Crypto Parameters in ipsec.conf 79
B.3.1 RSA SIG 80
B.3.2 PSK 81
B.4 Installing the OS and Openswan on Gateway GW2 82

Figures

1 High-Level Package Release Directory Structure 11
2 Development Board - Top View of Component and Connector Locations 15
3 Development Board - Bottom View of Component and Connector Locations 16
4 Development Board - Side View of the Board Connectors 17
5 Development Board System Setup 20
6 TDM Infrastructure with HSS Channel Driver Overview 59
7 Two analog voice mezzanine cards and one T1/E1 mezzanine card 60
8 Single analog voice mezzanine card 61
9 VPN Example 77

Tables

1 Development Board - Key Components and Connectors Legend 17
2 HSS Interrupt GPIO Pin Names 19

3 Output Files Created in the $ICP_BUILD_OUTPUT Directory 33
4 Pre-boot Firmware Setup Main Menu 52
5 Pre-boot Firmware Setup Program Action Keys 52
6 Serial Console Redirection Default Settings 53
7 Memory Allocation Settings 54
8 Linux Module/Driver Dependencies 57

Revision History

Date	Revision	Description
March 2009	003	Adding information on support for Asterisk®.
December 2008	002	Updated Section 3.5, "Unpacking the EP80579 Voice Software Linux Package" on page 26 to reflect the correct gzip names. Updated Section 4.1, "Environment Setup" on page 32 changing the commands to accomodate the scenario where the Linux* kernel sources are not installed in the standard location. Updated Section 4.4, "Optionally Installing Accelerated libSrtp" on page 34 to add note on accelerated SRTP option at end of section. Removed references to the software cryptolibrary (cryptosoft). Moved "Build and Install Openswan" and "Configuring Sample VPN Applications" chapters to Appendices at end of document. Updated Section A.3, "Building and Installing OpenSSL" on page 73. Removed references to a supplied Asterisk* channel driver. Other minor changes, marked with changebars.
September 2008	001	Initial release of this document.

§ §

1.0 Introduction

1.1 About this Manual

This Getting Started Guide documents the instructions to obtain, build (if necessary), install, and execute the Intel® EP80579 Software for IP Telephony Applications on Intel® QuickAssist Technology release package. Additionally, this document describes some brief instructions on configuring the Intel® EP80579 Integrated Processor with Intel® QuickAssist Technology Development Board.

Note: In this document, for convenience:

- The “Intel® EP80579 Integrated Processor with Intel® QuickAssist Technology Development Board” is referred to as the “development board”.
- The “Intel® EP80579 Software for IP Telephony Applications on Intel® QuickAssist Technology” is referred to as the “EP80579 IP Telephony software”.

Note: The release package includes a directory with sample code.

Note: Note that copying Linux* commands with quoted strings from this pdf file to a shell script may result in incorrect quotes. The user should edit the shell scripts appropriately.

1.2 Additional Information on Software

The EP80579 IP Telephony software release package has been validated with Red Hat* Enterprise Linux* 5.0.

1.2.1 Where to Find Current Software and Documentation

The software release and associated collateral can be found on the Hardware Design resource center.

1. In a web browser, go to http://www.intel.com/go/soc.
2. For Software and Pre-boot Firmware: Click on “Tools & Software” tab.
3. For Documentation: Click on “Technical Documents” tab.

Note: The EP80579 IP Telephony software release package contains encryption software and is subject to export requirements defined by the U.S Department of Commerce. To satisfy these requirements, the End User Certification Form must be filled out and submitted for review/approval. Instructions on this process are included during the download process. Please note that this process may take up to two business days to complete.

1.2.2 Product Documentation

The following documentation supporting this release can be accessed as described in Section 1.2.1:

- Intel® EP80579 Software for IP Telephony Applications on Intel® QuickAssist Technology for Linux* Getting Started Guide (this document)
- Intel® EP80579 Software for IP Telephony Applications on Intel® QuickAssist Technology Release Notes
- Intel® EP80579 Software for IP Telephony Applications on Intel® QuickAssist Technology Programmer's Guide
- Intel® EP80579 Software for IP Telephony Applications on Intel® QuickAssist Technology Linux* Device Driver API Reference Manual
- Intel® EP80579 Software for IP Telephony Applications on Intel® QuickAssist Technology TDM I/O Access API Reference Manual
- Intel® EP80579 Software on Intel® QuickAssist Technology Debug Services API Reference Manual
- Intel® EP80579 Software Drivers for Embedded Applications Programmer's Guide and API Reference Manual
- Software for Intel® EP80579 Integrated Processor Product Line PHY Porting Guide
- Intel® EP80579 Software for IP Telephony Applications on Intel® QuickAssist Technology - SRTP Acceleration Driver Application Note

1.2.3 Pre-boot Firmware

The latest release of the development board Pre-boot Firmware (BIOS) is also located on the Hardware Design resource center.

Please refer to Release Notes listed in Section 1.2.1 for the correct firmware version.

1.3 Related Software and Documentation

Refer to the Intel® EP80579 Integrated Processor with Intel® QuickAssist Technology Development Kit User's Guide for information on the development board including board layout, components, connectors, jumpers, headers, power and environmental requirements, and Pre-boot Firmware.

Please follow the directions in Section 1.2.1 to locate this collateral.

1.3.1 Open Source Software and Patches Required

Depending on your applications, the following table shows required open source software and patches.

Item	URL
OCF patch for Linux*	http://heanet.dl.sourceforge.net/sourceforge/ocf-linux/ocf-linux-26-20070727.patch.gz
OCF patch for OpenSSL*	http://heanet.dl.sourceforge.net/sourceforge/ocf-linux/ocf-linux-20080917.tar.gz
GNU Multiple Precision Arithmetic library	ftp://ftp.sunet.se/pub/gnu/gmp/gmp-4.2.1.tar.gz
Crypto Tools	http://heanet.dl.sourceforge.net/sourceforge/ocf-linux/crypto-tools-20070727.tar.gz
Openswan	http://www.openswan.org/download/old/openswan-2.4.9.tar.gz
Openswan Patch	http://heanet.dl.sourceforge.net/sourceforge/ocf-linux/ocf-openswan-2.4.9-20070727.patch.gz

Item	URL
Red Hat* Enterprise Linux Kernel	ftp://ftp.redhat.com/pub/redhat/linux/enterprise/5Client/en/os/SRPMS/kernel-2.6.18-8.el5.src.rpm
Open SSL	http://www.openssl.org/source/openssl-0.9.8g.tar.tar
LibsRTP	http://srtp.sourceforge.net/srtp-1.4.2.tgz

1.3.2 Intel® EP80579 Integrated Processor Software Shims

The software listed in this section contains sample source code provided “As-Is” without warranty of any kind. This software can be found in the Acceleration/shims folder of the software package provided by Intel:

- OCF Shim

1.4 Conventions

The following conventions are used in this manual:

- `Courier font` - commands and code examples

1.5 Software Overview

1.5.1 Features Implemented

The software provides the following features:

- Gigabit Ethernet (GbE) Controller Driver for Network Connectivity
- Support for Symmetric Cryptography (Synchronous and Asynchronous modes)
  - — Ciphers:
    - NULL Cipher
    - DES (ECB, CBC),
    - 3DES (ECB, CBC, CTR)
    - AES (ECB, CBC, CTR, CCM, GCM)
    - ARC4
  - — Hash Algorithms:
    - MD5
    - SHA-1
    - SHA-2 (224, 256, 384, 512)
    - Nested Hashing
  - — Authentication Schemes:
    - AES-XCBC-MAC-96
    - HMAC-MD-5
    - HMAC-SHA-1
    - HMAC-SHA-2 (224, 256, 384, 512)
  - — Chaining Cipher and Authentication algorithms
- Support for Public Key Cryptography (Synchronous and Asynchronous modes)
  - — Large Number Modular Exponentiation and Inversion
  - — RSA (Key Generation, Encrypt, Decrypt)
  - — DSA (Parameter Generation and Signatures)
  - — DH (Phase1 and Phase2 secret key generation)
  - — SSL/TLS Key and Mask generation

  - — True Random Number Generation (RNG)
  - — Prime Number Tests
- Support for OCF Shim (a kernel module that communicates with OCF at the top layer and the Intel® EP80579 Software for Security Applications on Intel® QuickAssist Technology Cryptographic API at the bottom layer and thus enabling OCF to use EP80579 Crypto Accelerators). The following features have been validated:
  - — Null Cipher
  - — DES-CBC, 3DES-CBC, AES-CBC, ARC4
  - — SHA1, SHA256, SHA384, SHA512, MD5 and HMAC of the same
  - — Modular Exponentiation and Chinese Remainder Theorem
  - — DSA RS Sign and Verify
  - — Diffie-Hellman secret key generation
  - — Daemon to supply random numbers to kernel /dev/random pool
- Institute of Electrical and Electronics Engineers (IEEE) 1588 Hardware Assist Driver
- Controller Area Network (CAN) Hardware Access Driver
- Advanced Host Controller Interface Software Support for Serial Advanced Technology Attachment (SATA) for Native Command Queuing and Hot Plug Capability
- System Management Bus (SMBus) Driver
- General Purpose I/O (GPIO) Hardware Access Driver
- Enhanced Direct Memory Access (EDMA) Hardware Assist Driver
- Watchdog Timer Hardware (WDT) Access Driver
- Support for three Time Division Multiplexing (TDM) ports, where each TDM port can be configured for up to four T1/E1 interfaces
- TDM clock signal that can be configured as an input/output port, or taken from an external reference clock
- Support for up to 128 channels, where each channel can be configured as an High-Level Data Link Control (HDLC) channel or a voice channel
- Support for up to four unidirectional channel bypasses, where a channel here refers to “a collection of timeslots”. All channel bypasses must be configured on the same TDM port. Currently, channel bypasses are only supported for one timeslot channels.
- Support for up to four Gain Control Tables

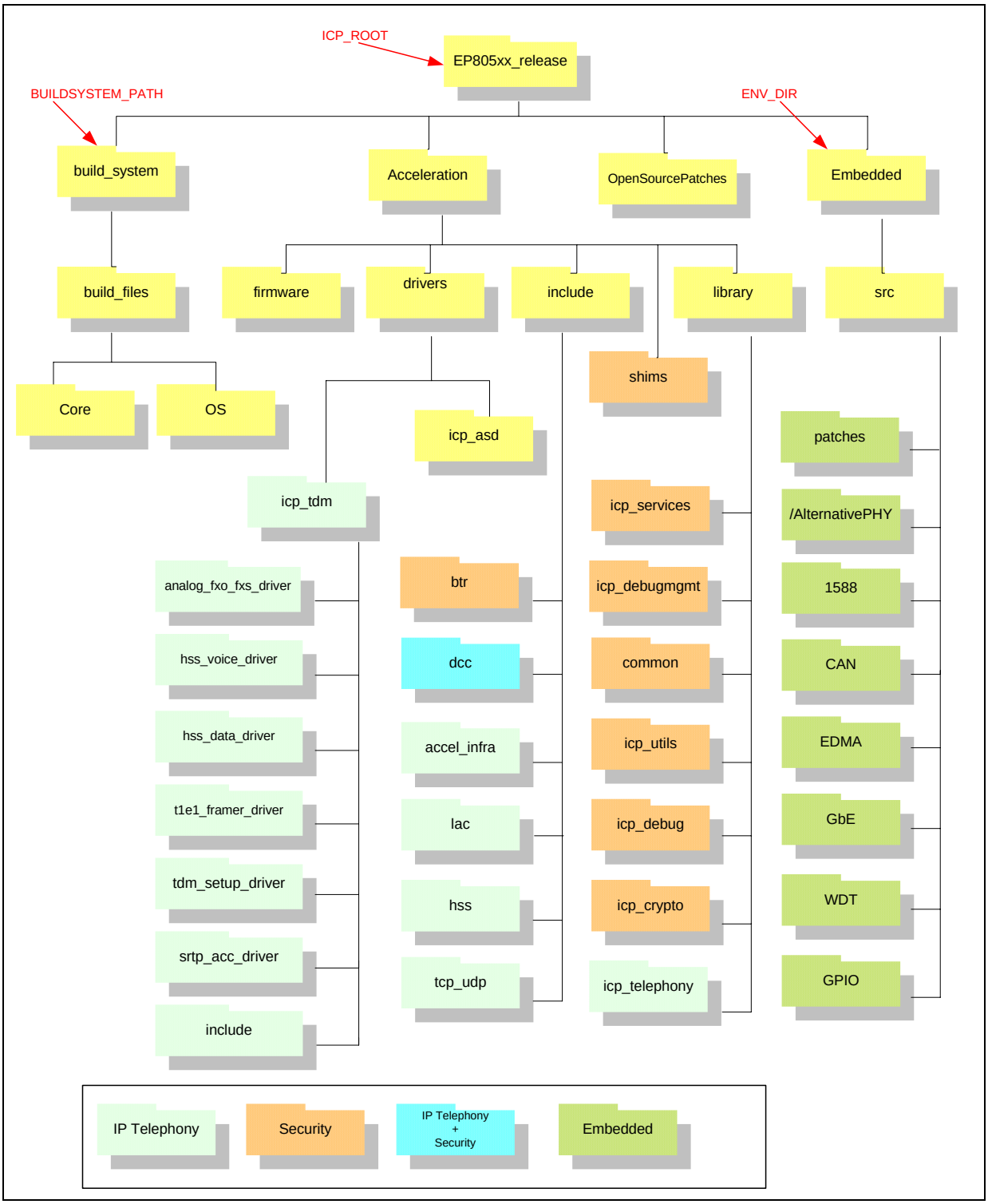
1.5.2 List of Files in Release

The Bill of Materials, sometimes referred to as the BOM, is included as a text file in the released software package. This text file is labeled “filelist” and is located at the top directory level for each release.

1.5.3 Package Release Structure

The high-level package release directory structure is shown in Figure 1:

Figure 1. High-Level Package Release Directory Structure



2.0 Configuration Requirements

2.1 Development Board Configuration

Complete details on the development board can be found in the Intel® EP80579 Integrated Processor with Intel® QuickAssist Technology Development Kit User's Guide. The document contains complete details on the design, structure, and function of all development board features.

To facilitate quick start of the Software for Intel® EP80579 Integrated Processor product line, relevant sections from the development kit User's Guide have been included in this chapter. Please follow the directions in Section 1.2.1 to access the full User's Guide.

2.1.1 Package Components

The development kit includes the following:

- A development board containing the Intel® EP80579 Integrated Processor with Intel® QuickAssist Technology
- ATX12V power supply
- One DDR2-800 Dual Inline Memory Module (DIMM)
- PCIe* graphics card
- SATA hard drive with cable
- SATA DVD-ROM with cable
- Two Controller Area Network cable connectors
- Power Cord (USA power cord supplied)

Customers requiring POTS or T1/E1 WAN connectivity should order these mezzanine cards in addition to the development kit. For additional information please visit http://www.intel.com/design/intarch/devkits/ep80579.

- Intel® EPAVM80579 Analog Voice Mezzanine Card
- Intel® EPTEM80579 T1/E1 Mezzanine Card

The following items are required, but not supplied by Intel:

- Mouse
- Keyboard
- Monitor
- Power Cord (if country or region-specific power cord is required)

Note: Additional items may be required but are not supplied by Intel.

2.1.2 Development Kit Setup

Ensure that all components listed in Section 2.1.1 arrive together. Once all components have been identified and located, installation and setup can begin. This section describes how to set up the development board for operation.

Note: This document assumes that the user is familiar with the basic concepts required to install and configure hardware for a PC system.

2.1.3 Safety

The development board is shipped as an open system allowing for maximum flexibility in changing hardware configurations and peripherals in a lab environment. Since the board is not in a protective chassis, the user is required to take safety precautions when handling and operating the board. Some assembly is required before use.

Ensure a safe and static-free work environment before removing any components from their anti-static packaging. The development board is susceptible to electrostatic discharge that may cause failure or unpredictable operation. The development board must be operated on a flame-retardant surface because a chassis is not included with the board.

Caution: Connecting the wrong cable or reversing a cable may damage the board and may damage the device being connected. Since the board is not in a protective chassis, use caution when connecting cables to the board.

Caution: The power supply cord provides the main connection to AC power. The socket outlet should be installed near the equipment and should be readily accessible. To avoid shock, ensure that the power cord is connected to a properly-wired and grounded receptacle. Do not connect/disconnect any cables or perform installation/maintenance of the boards in this product during an electrical storm. Ensure that any equipment to which this product will be attached is also connected to properly-wired and grounded receptacles.

Note: Ensure that the step to set up the ATX power supply is the final step performed in the process of assembly.

2.1.4 Connecting the Serial ATA Hard Drive and Cable

The development board provides two Serial ATA (SATA) connectors. Connect cables to the appropriate drive sequentially, starting from Port 0 to Port 1. See Figure 2 and Table 1 for the location and identification of the SATA connectors.

Note: Intel recommends connecting the boot drive to SATA port 0.

2.1.5 Connecting the Keyboard and Mouse

Connect a PS/2 mouse and keyboard to the stacked PS/2 connector on the rear panel of the board. The bottom connector is the keyboard connector and the top connector is the mouse connector. Alternatively, a USB keyboard and a USB mouse can be connected to the USB connectors on the development board.

Note: The mouse and keyboard are not supplied by Intel.

Note: The serial redirection feature can be enabled to remotely access the board through a serial cable without attaching a keyboard or mouse to the development board. Refer to the “Connecting the Serial Cable for Console Redirection” section of the Intel® EP80579

Integrated Processor with Intel® QuickAssist Technology Development Kit User's Guide for more information.

2.1.6 Connecting the PCI Express* Video Card

Populate the PCIe* graphics card in any one of the PCIe slots.

2.1.7 Connecting the Serial ATA DVD-ROM Drive (Optional)

Connect the Serial ATA DVD-ROM drive to SATA Port 1 utilizing the cable that comes with the DVD-ROM drive. See Figure 2 and Table 1 for the location and identification of the SATA connectors.

2.1.8 Connecting Voice and/or T1/E1 Mezzanine Card(s)

Depending on your application and software configuration, connect voice mezzanine card(s) and/or T1/E1 mezzanine card(s) to the 120-pin mezzanine connectors labeled EE, FF and KK (refer to Figure 2 and Figure 3).

There is no restriction on the order in which mezzanine cards can be installed on the development board. However, if you are also using the Intel® Accelerated DSP Software, connect mezzanine cards to the required slots as described in the "Test Setup" section of the Release Notes delivered as part of the Intel® Accelerated DSP Software release. If the application includes voice and data, the sequence of the voice and T1/E1 mezzanine cards is dependent on the software configuration.

2.1.9 Connecting the Power Cables

Use the following procedure to connect the power cables:

1. The board supports the use of ATX12V power supplies with either 2 x 10 or 2 x 12 main power cables.
2. Plug the main connector into the board. Ensure that the plug clip lines up with the clip lock and the connector pins easily fit into their appropriate slots. When using a power supply with a 2 x 10 main power cable, attach that cable to the right-most part of the main power connector, leaving pins 11, 12, 23 and 24 (labeled on the board) unconnected.
3. Plug in the power connectors from each of the SATA drives.

2.1.10 Powering Up the System

Warning: Ensure the steps in the previous sections were strictly followed before powering up the system.

Use the following procedure to power up the development board:

1. Ensure that the processor heat sink and the fan are installed according to the procedure in the "Connecting the Processor Heatsink and Fan" section of the Intel® EP80579 Integrated Processor with Intel® QuickAssist Technology Development Kit User's Guide.
2. Leaving the On/Off switch in the OFF position, connect the power cable into the back of the power supply.
3. Once the board is set up, plug the cord into the power source.
4. Switch on the power supply.
5. Press the power-on button to start the system. Refer to Figure 2 for the location of the power-on button (item I, lower-left).

Note: Table 1 is a legend for key items labeled in Figure 2, Figure 3 and Figure 4.

Figure 2. Development Board - Top View of Component and Connector Locations

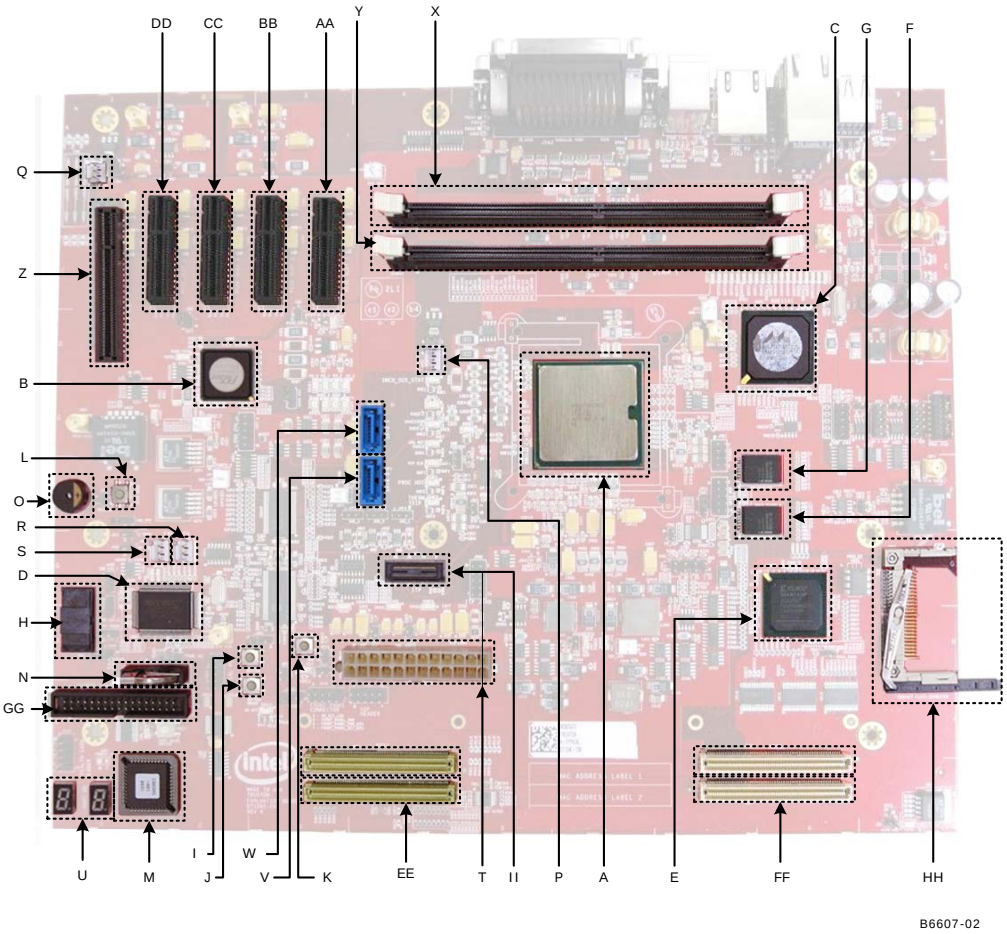


Figure 3. Development Board - Bottom View of Component and Connector Locations

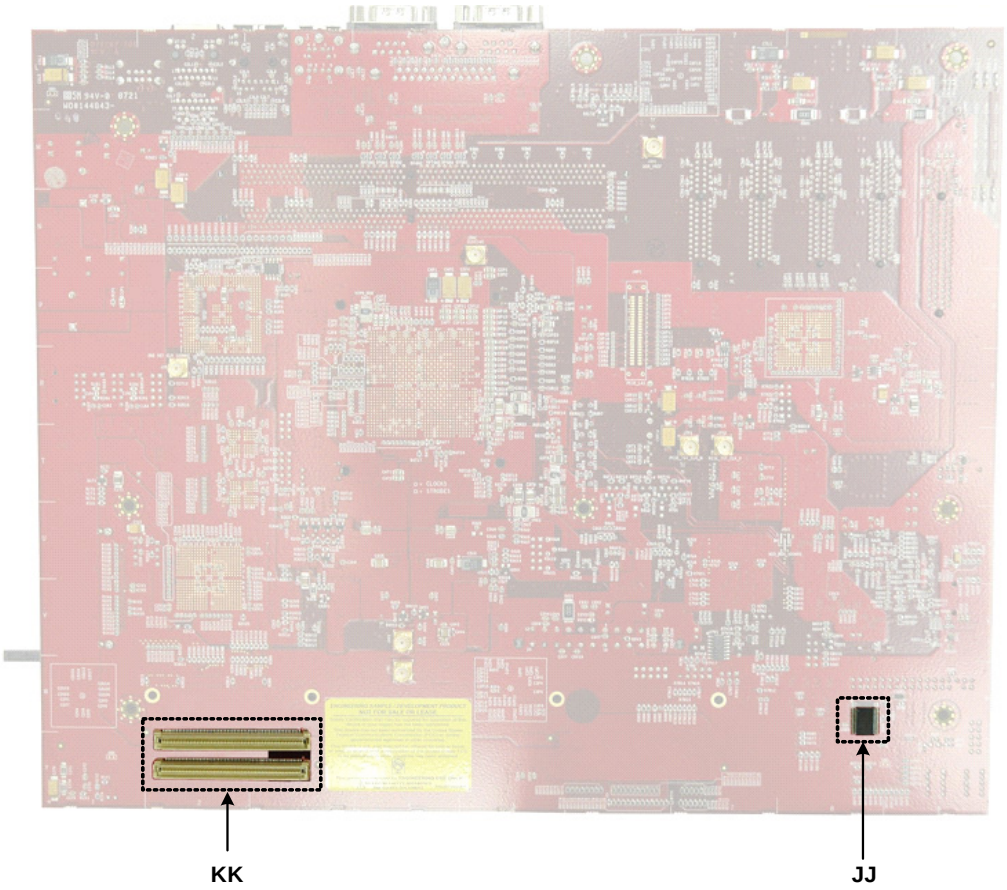
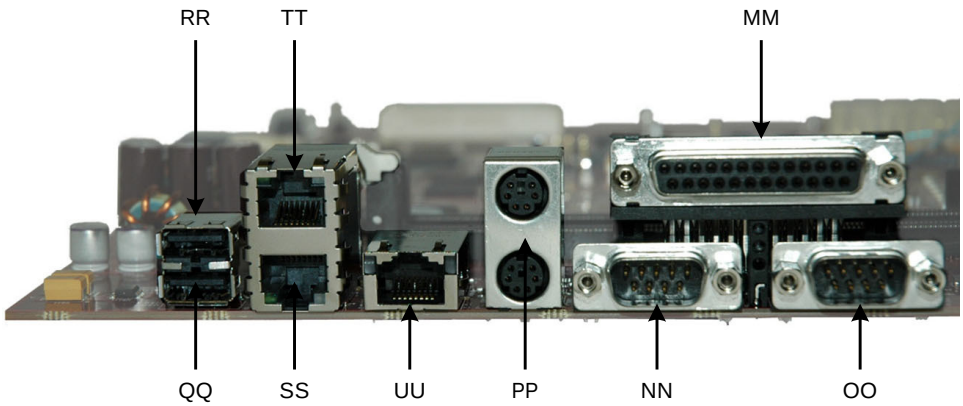


Figure 4. Development Board - Side View of the Board Connectors



B6605-01

Table 1. Development Board - Key Components and Connectors Legend

Callout	Component/Connector
A	Intel® EP80579 Integrated Processor with Intel® QuickAssist Technology
B	PEX PCIe Switch Chip
C	Marvell* 88ME1141 Quad PHY
D	Super IO Controller
E	FPGA
F	Flash memory 0
G	Flash memory 1
H	Firmware Hub(FWH)
I	Power button
J	Reset button
K	Sleep button
L	PCIe Wake button
M	Port 80 IC
N	CMOS battery
O	On-board speaker
P	CPU FAN connector
Q	AUX FAN connector
R	AUX0 FAN connector
S	AUX1 FAN connector
T	ATX power connector
U	Two 7-segment display (Port 80)
V	SATA port 0

Table 1. Development Board - Key Components and Connectors Legend

Callout	Component/Connector
W	SATA port 1
X	DDR2 DIMM0
Y	DDR2 DIMM1
Z	Slot 0 x8 connector 4 lanes PCI Express
AA	Slot 1 x4 connector 1 lane PCI Express
BB	Slot 2 x4 connector 1 lane PCI Express
CC	Slot 3 x4 connector 1 lane PCI Express
DD	Slot 4 x4 connector 1 lane PCI Express
EE	Mezzanine connector 1
FF	Mezzanine connector 0
GG	Floppy Connector
HH	CF connector
II	ITP-XDP connector
JJ	Trusted Platform Module
KK	Mezzanine connector 2
MM	Parallel port
NN	COM1
OO	COM2
PP	PS/2 mouse (top)/keyboard (bottom)
QQ	USB port 0
RR	USB port 1
SS	RJ-45 Ethernet port 0
TT	RJ-45 Ethernet port 1
UU	RJ-45 Ethernet port 2

2.2 Development Platform Configuration for IP Telephony Applications

The development board includes three sets of mezzanine connectors dedicated to HSS devices. There are two primary cards intended for use with the development board, an analog voice card (FXS/FXO) and a quad T1/E1 card. The two cards are different with respect to the offset of the mezzanine connectors; one is offset left, the other is offset right.

Each mezzanine card has two 120-pin connectors associated with it. The first connector is referred to as the standard connector. The second connector is referred to as the expansion connector. The standard connector includes primary and secondary HSS interfaces, expansion bus, GPIOs and interrupt. The expansion connector includes SSP signals, extra GPIOs and eight expansion bus chip select signals.

The development board uses three GPIO pins to provide interrupts on the HSS mezzanine connectors. Table 2 shows the GPIO pins that are used as HSS interrupts.

Table 2. HSS Interrupt GPIO Pin Names

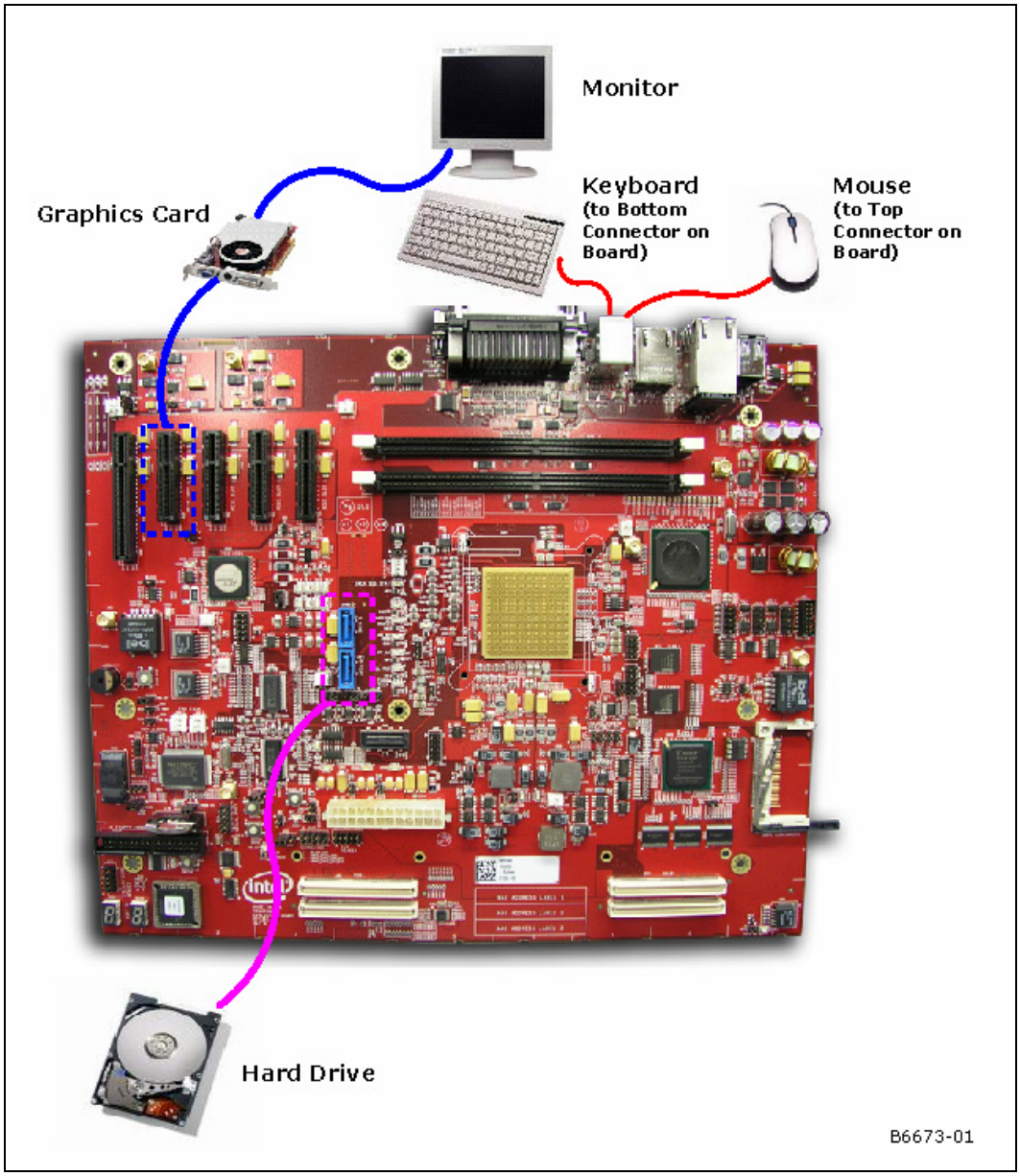
Pin Name	HSS Interrupt Number
GP16_IRQ24	HSS0_INT_OUT
GP17_IRQ25	HSS1_INT_OUT
GP18_IRQ36	HSS2_INT_OUT

2.3 Development Board Setup Requirements

Figure 5 shows the system setup when the target development board is also used for build and install.

Note: If you are downloading Intel Performance Primitives (IPP), connect the eth0 port to the internet. This is required since verification is done over the internet. If you forget to connect eth0 initially and later want to connect and enable IPP, use the “service network restart” command after connecting eth0 to the internet.

Figure 5. Development Board System Setup



3.0 Installing the OS on a Development Board

Installing the OS on a development board involves the installation of Red Hat* Enterprise Linux* 5.0 (RHEL 5.0) from the CDs and the rebuilding of the kernel and module files after patching the kernel for the following:

- Built-in PCI device recognition
- OpenBSD Cryptographic Framework (OCF) patch

Note: System commands given in this chapter assume that the user is issuing commands from a bash shell. This is the default shell. Use the “echo $0” command to verify use of the bash shell or run “/bin/bash” to switch to the bash shell.

3.1 System Requirements

Please consult the Red Hat Linux minimum hardware requirements in the online Installation Guide at http://www.redhat.com/docs/manuals/enterprise/RHEL-5-manual/Installation_Guide-en-US/index.html. The development board meets these requirements.

Execute the following three steps:

- Get the latest BIOS image and install it as described in Section 9.2.1, “Aptio Flash Update Utility (AFUEFI)” on page 55
- Set the Legacy or Advanced Host Controller Interface (AHCI) SATA mode as described in Section 9.1.5, “Legacy and AHCI SATA Mode” on page 54
- Allocate the Coherent and Non-Coherent Memory as described in Section 9.1.4, “Coherent and Non-Coherent Memory Allocation” on page 54

Note: The Legacy/ AHCI SATA mode cannot be changed after OS installation.

3.2 Acquiring Red Hat Enterprise Linux 5.0

The software package has been validated with Red Hat Enterprise Linux 5.0, and validated with GCC 4.1.1 and the Glibc 2.5 tool chain. The software package does not include a distribution of Red Hat Enterprise Linux 5.0 or any OS. The package includes Linux device driver sources developed by Intel for EP80579 integrated processor features. The device driver sources may be installed with a Red Hat Enterprise Linux 5.0 distribution. Please acquire a source distribution of Red Hat Enterprise Linux 5.0 either from Red Hat or the distributor of your choice.

3.3 Installing Red Hat Enterprise Linux 5.0

For complete Red Hat Enterprise Linux 5.0 installation instructions, please refer to the online Installation Guide at http://www.redhat.com/docs//manuals/enterprise/RHEL-5-manual/Installation_Guide-en-US/index.html. The following are some basic instructions. For the purposes of this Getting Started Guide, it is assumed that the installation is from CD images.

Note: Red Hat Enterprise Linux 5.0 is available in Server and Client packages; only the Client package is considered in this guide.

1. Prepare the system to boot from CDROM by using the boot selections in the BIOS Setup Menu. Details of the BIOS setup options are available in Chapter 9.0, "Pre-boot (BIOS) Firmware". In the EFI shell, choose the BOOT option, then set the boot device order so that the CD Drive is the first boot option.
2. Power on the system with disc #1 in the CDROM. The system should begin to boot from the Red Hat installation disc #1.

Note: It is recommended that all Red Hat installation discs be verified at least once prior to an installation of the OS.

3. Upon booting from disc #1, the installation process prompts if the CD media should be tested.
4. Review the Red Hat Enterprise Linux 5 Release Notes during the installation if desired. Continue with the installation process whenever prompted with the "Next" button.
5. Select the language to use during the Red Hat Enterprise Linux 5.0 installation process.
6. Select the appropriate keyboard for the system.
7. When prompted, enter the Installation Number (also known as the License Key), and press "OK".
8. Select the option to perform a fresh install of Red Hat Enterprise Linux on the system.
9. The hard drive that comes with the development board is initially unformatted. Partition this hard drive.
10. Select the time zone in which the system is located.
11. Select the city nearest to the system's located.
12. Enter the desired root password for the system in the box labeled "Root Password:" and reaffirm the root password by entering the password in the box labelled "Confirm:".
13. The installer prompts for office or multimedia default setups, both of these should be "unchecked".

Note: If you skipped the input of the installation number (step 7.), you may not see the "Software Development" software set indicated in step 14.

IMPORTANT: You must select the "Customize Now" button in step 14. to get access to the "Software Development" options required to complete the installation successfully.

14. Select "Customize Now" and see Section 3.3.1, which provides "Development" options that enable GNU Compiler Collection (Gcc) and GlibC Tool Set installation. Select the desired sets of software to include. Selecting the "Software Development" software set ensures that Gcc and GLibC Tool Set are installed. These are necessary to allow the compilation of the EP80579 security software, EP80579 embedded software drivers and the rebuilding of the Linux kernel to support the EP80579 integrated processor.

For application development, for example, using Asterisk®, the EP80579 HSS channel driver build/install requires libtiff-devel (libtiff-devel-3.8.2-7.el5.i386.rpm) in addition to gcc-c++ (gcc-c++-4.1.1.52.el5.i386.rpm) and libstdc++-devel (libstdc++-devel-4.1.1-52.el5.i386.rpm).

libtiff-devel (libtiff-devel-3.8.2-7.el5.i386.rpm) for example may not be part of the installation CD. This needs to installed separately.

3.3.1 Recommended Installation Customizations

1. Package customization is available through these selections:
   - — Desktop Environments
   - — Applications
   - — Development
   - — Servers
   - — Base System
   - — Languages

   The recommended minimum package installation selections are given in the tips below. Additional detailed customization of each package can be accomplished by selecting a package (for example: Editor), and clicking the “Optional package” button to display the optional and default packages within the main package.

Tip: **Desktop Environments:** Optional. Install Gnome or KDE if you wish to use a Graphical User Interface (GUI).

Tip: **Applications:** Editor. Others are optional.

Tip: **Development:** Development Libraries and Development Tools are recommended to be installed. Others are optional.

Under Development, in addition to the default selection, you must select the LEGACY SOFTWARE DEVELOPMENT package for c++ development. This is necessary for correct operation of the sample application that uses Asterisk which is described in this Getting Started Guide.

Please note that LEGACY SOFTWARE DEVELOPMENT comes with all the optional packages selected by default: gcc, gcc-c++, gcc-Fortran 77, glibc standard library, glibstdc++ standard libraries, and glibc++ RH standard libraries.

Be sure that the optional packages for C and C++ are all selected. You can select the LEGACY SOFTWARE DEVEOPMENT option and click “Optional packages” at the bottom right side of the screen. A popup window shows the components. Please verify that they are selected. If not, select the gcc, gcc-c++, glibc, libstdc++ standard c++ libraries and libstdc++ RH standard c++ libraries.

For your convenience, the full names of the above optional packages for the LEGACY SOFTWARE DEVELOPMENT option are shown below:

- compat-gcc-34 -3.4.6-4.I386 - Compatibility GNU Compiler Collection
- compat-gcc-34 -c++ - 3.4.6-4.I386 - C++ support for compatibility compiler
- compat-gcc-34-g77 -3.4.6-4.I386 - Fortran 77 support for compatibility compiler
- compat-glibc -1:2:3.4-2.26.I386 - Compatibility C library
- compat-libstdc++-296 - 2.96-138.I386 - Compatibility 2.96-RH Standard C++ libraries
- compat-libstdc++-33 -3.2.3-61.I386 - Compatibility standard C++ libraries

Tip: **Servers:** Depending on the end use connectivity, none to all server types may apply. Some common selections are DNS Name Server, FTP Server, Legacy Network Server, Network Servers or Web Servers.

Tip: **Base System:** Administration Tools, Base and System Tools are recommended to be installed.

Tip: **Languages:** Additional language selections vary depending on user location.

2. Following package selection, click the “Next” button when prompted. After a dependency check, the installation process is ready to begin and prompts the user to have the required Red Hat Enterprise Linux CDs available. To begin the installation, click the “Continue” button, click “Back” to change a package selection, or click “Reboot” to abort this installation.
3. If continuing with this installation, when prompted, insert the requested installation CD and click “Next”. When the installation is complete, and a prompt is displayed to reboot the system, remove the final CD from the CDROM and choose to reboot the system.
4. When rebooting, some normal system setup is required, such as setting the time and date. When asked, follow the prompt by clicking the “Forward” button.
5. Disable “Firewall”.

Note: If the Firewall is not disabled, the customer would have to execute appropriate iptable commands to allow VPN traffic.

6. SELinux should be changed from the default “Enforcing” to “disabled”.
7. Enabling kdump is optional, but may help to diagnose the cause of any future system crash. Minimum memory allocated to kdump is 128 MB. If kdump is selected, a reboot is required.
8. Set the time and date.
9. Connecting to RHN is optional and dependent upon having network connectivity.

Note: Although you are creating additional users in the next step, please note that the instructions in this document should be executed as the root user.

10. Create additional users.
11. Finally, when prompted, press the “Finish” button.

3.4 Installing Red Hat Enterprise Linux 5.0 Kernel Source

Note: Always login as the root user to carry out the instructions in this document.

1. Follow the instructions in the man page for the “date” command to set the current date. For example:

   ```
   date 072910302008
   ```

   to set the date to July 29th, 2008, 10:30AM.

Note: If the Time, Date and Timezone are not set on the development board, the build command used later in this procedure will go into an infinite loop.

2. Create a directory for file download, build and install activities. For example, create a directory called “EP805XX_release” with the following commands:

   ```
   mkdir /EP805XX_release
   cd /EP805XX_release
   ```

3. Create the ICP_ROOT environment variable:

   ```
   export ICP_ROOT=$PWD
   ```

Note: Before installing any of the rpms referenced in the next step, you can use the `rpm -q <package_name>` command (where, <package_name> = elfutils,

ncurses-devel, rpm-build, or unifdef) to check if the corresponding package is already installed.

4. Copy the elfutils, ncurses-devel, rpm-build (OS Disk 5, Workstation directory) and unifdef (OS Disk6, Workstation directory) rpms from the RHEL 5.0 Client Installation CDs to the $ICP_ROOT directory and install the rpms using the following commands if they are not already installed.

Note: The command 'rpm -q <package_name>' can be used to check if the package has already been installed. For example, 'rpm -q elfutils' will check if the package elfutils-0.125-3.el5.i386.rpm has been installed.

a. Mount the CD-ROM drive, for example:

```
mkdir /mnt/cdrom
mount -t auto /dev/cdrom /mnt/cdrom
```

b. With OS Disk 5 inserted in the CD-ROM drive, issue the following commands:

```
cp /mnt/cdrom/Workstation/elfutils-0.125-3.el5.i386.rpm .
cp /mnt/cdrom/Workstation/rpm-build-4.4.2-37.el5.i386.rpm .
cp /mnt/cdrom/Workstation/ncurses-devel-5.5.-24.20060715.i386.rpm .
```

c. Unmount the CD-ROM drive before removing OS Disk 5 and inserting Disk 6 as follows:

```
unmount /mnt/cdrom
```

d. Remove OS Disk 5 and insert Disk 6 into the CD-ROM drive.

e. Mount the CD-ROM drive:

```
mount -t auto /dev/cdrom /mnt/cdrom
```

f. Copy the file from OS Disk 6 by issuing the following command:

```
cp /mnt/cdrom/Workstation/unifdef-1.171-5.fc6.i386.rpm .
```

g. Unmount the CD-ROM drive:

```
umount /mnt/cdrom
```

h. Install the rpms as follows:

```
rpm -ivh elfutils-0.125-3.el5.i386.rpm
rpm -ivh rpm-build-4.4.2-37.el5.i386.rpm
rpm -ivh ncurses-devel-5.5-24.20060715.i386.rpm
rpm -ivh unifdef-1.171-5.fc6.i386.rpm
```

5. If the eth0 port is connected to the internet, you can ftp the rpm given below directly from the target machine. If not, proceed as follows:

   Use ftp from another machine to get the kernel source rpm packages:

   ```
   ftp://ftp.redhat.com/pub/redhat/linux/enterprise/5Client/en/os/SRPMS/kernel-
       2.6.18-8.el5.src.rpm
   ```

   Using a Flash drive or CDROM, copy the kernel source rpm package to the $ICP_ROOT directory on the target machine.

   A Flash drive can be mounted as follows:

```
mkdir /mnt/sdxx
mount /dev/sdxx /mnt/sdxx
```

Copy the kernel source rpm using the following command:

```
cp /mnt/sdxx/kernel-2.6.18-8.e15.src.rpm $ICP_ROOT
```

Execute the following commands:

```
cd $ICP_ROOT
mkdir -p /usr/src/redhat // if not already created
rpm -ivh kernel-2.6.18-8.el5.src.rpm --nosignature
```

Note: If the “warning: user brewbuilder does not exist - using root” and “warning: group brewbuilder does not exist - using root” message is displayed, it can be ignored.

The RPM .spec file (/usr/src/redhat/SPECS/kernel-2.6.spec) is created. This spec file must be installed using the rpmbuild command as indicated in the following step.

Note: In the following step, although messages are displayed immediately after the command is entered, there may be delays between messages. Please wait until the prompt is displayed.

6. Execute the following commands:

   ```
   cd /usr/src/redhat/SPECS
   rpmbuild -bp kernel-2.6.spec
   ```

   A new /usr/src/redhat/BUILD/kernel-2.6.18 source directory is created.

7. Set the environment variable KERNEL_SOURCE_ROOT to the new kernel source directory:

   ```
   export KERNEL_SOURCE_ROOT=/usr/src/redhat/BUILD/kernel-2.6.18/linux-2.6.18.i386
   ```

8. Also, set the following symbolic link to the kernel source directory:

   ```
   mkdir /usr/src/kernels    /* if not already created*/
   ln -s $KERNEL_SOURCE_ROOT /usr/src/kernels/linux
   ```

3.5 Unpacking the EP80579 Voice Software Linux Package

The EP80579 IP Telephony software package comes in the form of a tarball. See Section 1.2.1, “Where to Find Current Software and Documentation” on page 7 for the software location. The package can be unpacked at any location on the system, but for the purposes of this Getting Started Guide, a recommendation is provided.

Transfer the gzip file to the EP80579 target system using any preferred method, for example, a USB flash drive, CDROM or network transfer. Unpack the tarball in the $ICP_ROOT directory using the following commands:

Note: Without the option ‘m’ in the following tar command, the build might get into an infinite loop if the Date is set incorrectly.

```
tar -xmzvf Telephony.L.1.x.x-xx.tar.gz
```

Two tarball files are created; one for the software package and one for Open Source patch files. Unpack the software package tarball in the /EP805XX_release directory using the following command:

```
tar -xmzvf Telephony.L.1.x.x-xx_SW.tar.gz
```

A new directory structure is created under the /EP805XX_release directory that contains the EP80579 Linux software. See Section 1.5.3, “Package Release Structure” on page 10 for details of the directory structure of the software release.

The /Install directory is also created under the /EP805XX_release directory. The /Install directory contains scripts for installing and configuring drivers.

You will also see a file “filelist” that contains the list of files included in the release.

Unzip the Open Source patch files. These files are needed later in Section 3.7.3, “Backporting OCF Features and Bug Fixes” on page 29.

```
tar –xmzvf PatchFiles_Telephony.L.1.x.x-xx.tar.gz
```

3.6 Patching the Kernel for PCI Device Recognition

The EP80579 IP Telephony software package includes two patch files that must be applied to the Linux kernel. The released patch files, named “Intel_EP80579_RHEL5.patch” and “pci.ids_RHEL5.patch” are found in the $ICP_ROOT/Embedded/src/patches directory.

1. Execute the following commands to move the patch:

   ```
   cd /usr/src/redhat/BUILD/kernel-2.6.18
   cp $ICP_ROOT/Embedded/src/patches/Intel_EP80579_RHEL5.patch .
   ```

2. Apply the patch to the unpacked kernel source using the following command:

   ```
   patch -p0 < Intel_EP80579_RHEL5.patch
   ```

   When the patch is applied successfully, the output is as follows:

   ```
   patching file linux-2.6.18.i386/arch/i386/pci/irq.c
   patching file linux-2.6.18.i386/drivers/scsi/ahci.c
   patching file linux-2.6.18.i386/drivers/scsi/ata_piix.c
   patching file linux-2.6.18.i386/drivers/i2c/busses/Kconfig
   patching file linux-2.6.18.i386/drivers/i2c/busses/i2c-i801.c
   patching file linux-2.6.18.i386/include/linux/pci_ids.h
   patching file linux-2.6.18.i386/arch/i386/kernel/cpu/intel_cachinfo.c
   ```

The following patch must be applied to ensure the correct device strings are returned by the lspci utility.

Change directory to the */usr/share/hwdata* directory and place the pci.ids_RHEL5.patch in this directory. Apply the patch to the pci.ids file using the following command:

```
cd /usr/share/hwdata
cp $ICP_ROOT/Embedded/src/patches/pci.ids_RHEL5.patch .
patch -p0 < pci.ids_RHEL5.patch
```

Upon patch application, a positive resulting output looks like the following:

```
Patching file pci.ids
```

3.7 (Optional) Modifications for Use with OCF

The modifications described in this section are required only for use with the OpenBSD Cryptographic Framework (OCF).

3.7.1 Applying the OCF Patch

The source for this patch can be obtained from the following URL:
http://heanet.dl.sourceforge.net/sourceforge/ocf-linux/ocf-linux-26-20070727.patch.gz

1. Download the patch to the $ICP_ROOT directory and gunzip it as follows:

```
cd $ICP_ROOT
gunzip ocf-linux-26-20070727.patch.gz
cd $KERNEL_SOURCE_ROOT
patch -p1 < $ICP_ROOT/ocf-linux-26-20070727.patch
```

2. This patch creates the crypto/ocf folders and modifies a couple of kernel source files. However, it does not modify crypto/Kconfig and crypto/Makefile. These modifications need to be applied manually using the following commands:

   a) Change the directory

```
cd $KERNEL_SOURCE_ROOT/crypto
```

   b) Add the line ‘source “crypto/ocf/Kconfig”’ at line 364:

```
sed -i 364i"source\ \"crypto/ocf/Kconfig\"" Kconfig
```

   c) Append the line ‘obj-$(CONFIG_OCF_OCF) +=ocf/’

```
echo "obj-\$(CONFIG_OCF_OCF) += ocf/" >> Makefile
```

3.7.2 Configuring OCF Parameters

1. Make backup copies of the default configuration file .config and execute the following commands to replace the default configuration file:

```
cd $KERNEL_SOURCE_ROOT
make mrproper
cp /boot/config-2.6.18-8.el5 .config
sed -i s/2.6.18-8.el5/2.6.18-EP805XX/ .config
```

2. Edit the .config file is:

```
make menuconfig
```

3. In the graphical interface, navigate to “Cryptographic Options”.
4. Under “OCF Configuration”, set “OCF (Open Cryptographic Framework)” to ‘M’ ('M' sets the feature to be built as a module).
5. The above setting spawns, a larger menu. Set the following entries to ‘M to enable the random number generation daemon’:

```
“crypto random --- harvest entropy for /dev/random”
```

6. The above entry spawns a sub-menu. Verify that the following entry is blank to stop additional checks for randomness.

   ```
   "enable fips RNG checks"
   ```

7. Set the following entry to 'M' to provide support for OCF invocation from user space (cryptodev).

   ```
   "cryptodev (user space support)"
   ```

8. Ensure ocf-bench is set to blank.

   ```
   "ocf-bench (HW crypto in-kernel benchmark)"
   ```

9. Exit the graphical interface by pressing the Escape button several times. Save the new configuration.

Note: Double check that FIPS_TEST_RNG is not defined in OCF. The Lookaside Crypto (LAC) component provides a true random number and these checks are not required on the date provided.

3.7.2.1 Copying Include Files for OCF

Execute the following commands to copy the required include files:

```
mkdir -p /usr/include/crypto
cp $KERNEL_SOURCE_ROOT/crypto/ocf/cryptodev.h /usr/include/crypto/
```

3.7.3 Backporting OCF Features and Bug Fixes

OCF EP80579 driver has been tested for both OCF 20070727 and OCF 20071215. Since version OCF 20071215 in combination with Openswan 2.4.11 has a performance issue for high rates of traffic, the recommended versions are OCF 20070727 and Openswan 2.4.9. The following patches fix errors found in OCF20070727 and backport functionary present in OCF 20071215. Apply the two patches using the following commands:

```
cd $KERNEL_SOURCE_ROOT
patch -p0 < $ICP_ROOT/OpenSourcePatches/linux-ocf-20070727-backport.patch
patch -p0 < $ICP_ROOT/OpenSourcePatches/ocf-linux-20070727-driver-removal.patch
```

3.8 Rebuilding the Kernel

Note: Do NOT execute the commands in Section 3.8.1 if you executed commands in Section 3.7, "(Optional) Modifications for Use with OCF" on page 28. In this case, continue with Section 3.8.2.

Note: A spurious interrupt on IRQ 169 can be prevented by ensuring the Logical Volume Manager (LVM) is not enabled during Linux kernel configuration (at compile time). However, if LVM is enabled, the spurious interrupt will not affect system performance and will eventually be disabled by the Linux kernel with no adverse effect.

3.8.1 Restore the Old Configuration

Note: When the user issues the following commands, the kernel config file is stepped on, OCF has to be turned on again, and the kernel has to be renamed to 2.6.18-EP805XX.

```
cd $KERNEL_SOURCE_ROOT
```

```
make mrproper
make oldconfig
```

3.8.2 Modifying the Makefile

Change the directory using the following command:

```
cd $KERNEL_SOURCE_ROOT
```

Modify the Makefile file in the current directory. Edit the line “EXTRAVERSION = -prep” by editing “-prep” to a name meaningful to the purpose of recompiling the kernel. A simple suggestion is to change “-prep” to “-EP805XX”. Once the kernel is recompiled, this helps to indicate which vmlinuz kernel files and entries in the /etc/grub.conf file are related to the recompiled kernel.

3.8.3 Building and Installing the Patched Kernel Source

Change to the directory $KERNEL_SOURCE_ROOT and execute the following commands to recompile the Linux kernel:

```
make && make modules_install && make install
```

Note: This step will take about an hour and a half to complete.

Save the configuration of the new 2.6.18-EP805XX kernel in the /boot directory using the following command:

```
cp $KERNEL_SOURCE_ROOT/.config /boot/config-2.6.18-EP805XX
```

Note: The make commands will be executed only if the preceding make commands are successful.

The new kernel is compiled and placed in the /boot directory. The new kernel is named vmlinuz-2.6.18-EP805XX. A symbolic link is created from vmlinuz -> vmlinuz-2.6.18-EP805XX.

3.8.4 Making the New Kernel Default

Also, the /boot/grub/grub.conf file is modified to include the possibility of booting to this newly compiled kernel. An entry is created for the new kernel:

```
title Red Hat Enterprise Linux Client (2.6.18-EP805XX)
    root (hd0,0)
    kernel /vmlinuz-2.6.18-EP805XX ro root=/dev/VolGroup00/LogVol00 rhgb quiet
    initrd /initrd-2.6.18-EP805XX.img
```

1. The title may be changed to be more reflective of this kernel’s purpose or name.
2. To boot to this kernel image by default, modify the /boot/grub/grub.conf file’s “default” entry. Entries begin from zero (0) and are counted top-down. Edit “default=<kernel entry number>”, where <kernel entry number> is the number entry for the new vmlinuz-2.6.18-EP805XX kernel.
3. Skip to step (5) if your platform uses less than 1GB of memory.
4. Add the “ mem=832M” (without the quotes, use leading space to demarcate) to the end of the kernel line. A sample entry might look like the following assuming 1 GB physical memory.

```
kernel /vmlinuz-2.6.18-EP805XX ro root=/dev/VolGroup00/LogVol00 rhgb mem=832M
```

Note: 832M = 1G - 32M(NCDRAM size) - 32M(CDRAM size) -128M).
32M = 0x2000 (BIOS setting for CDRAM and NCDRAM size) * 4K (page size)

Although we do not require 32M of Coherent DRAM (CDRAM) and Non-Coherent DRAM (NCDRAM) for this release, all validations have been done using the above values. All features have been validated using these BIOS settings.

5. Save the changes.

3.8.5 Rebooting and Verifying

Rebooting the system reboots to the newly compiled and installed kernel.

Execute the following command to ensure that the correct kernel has been loaded:

```
uname -r
```

The output should look like:

```
2.6.18-EP805XX
```

4.0 Building EP80579 IP Telephony Software on a Target Development Board

This chapter provides instructions for unpacking the EP80579 IP Telephony software package, setting up the environment and building/compilation instructions.

Note: System commands given in this chapter assume that the user is issuing commands from a bash shell. This is the default shell. Use the “echo $0” command to verify use of the bash shell or run “/bin/bash” to switch to the bash shell.

4.1 Environment Setup

After unpacking the release package (as described in Section 3.5), issue the following commands to restore the environment variables.:

Note: All environment variables set for kernel rebuild are lost when the system was rebooted.

Note: As per the instruction in the previous chapters, the kernel source are untarred at KERNEL_SOURCE_ROOT and other sources are untarred at ICP_ROOT.

```
export ICP_ROOT=/EP805XX_release
export ICP_BUILDSYSTEM_PATH=$ICP_ROOT/build_system
export KERNEL_SOURCE_ROOT=/usr/src/redhat/BUILD/kernel-2.6.18/linux-2.6.18.i386
/bin/mkdir -p /usr/src/kernels
/bin/ln -s $KERNEL_SOURCE_ROOT /usr/src/kernels/linux
```

Follow the instructions in the man page for the “date” command to set the current date. For example,

```
date 102910302007
```

sets the date to October 29th, 2007, 10:30AM.

4.2 Build Options

Section 4.3, “Build Using Top Level Make” on page 33 describes the instructions for building IP Telephony, Security and Embedded components together. The OCF Shim component is always built separately and is optional for IP Telephony applications.

Refer to Chapter 6.0, “Building Security and Embedded Components Individually” to install various security and driver components individually.

See Section 7.6.4, “Alternative PHY Compilation” on page 46 for information on compiling the Gigabit Ethernet driver for the National Semiconductor* DP838481 PHY. See the Software for Intel® EP80579 Integrated Processor Product Line PHY Porting Guide for information on porting the driver to other PHYs.

4.3 Build Using Top Level Make

It is required to set an ICP_BUILD_OUTPUT environment variable that defines where built objects are copied and where the sample load scripts can find the built objects.

For example, to store the resulting files in a directory called $ICP_ROOT/StagingArea, use the following commands:

```
mkdir $ICP_ROOT/StagingArea
export ICP_BUILD_OUTPUT=$ICP_ROOT/StagingArea
```

The use of the export command as shown in Section 4.1, “Environment Setup” on page 32 to export ICP_ROOT, ICP_BUILDSYSTEM_PATH, KERNEL_SOURCE_ROOT and ICP_BUILD_OUTPUT can be entered in a script file and appended to the bash-profile bash shell startup file, so that every time the system is started, these environment variables are set automatically. However, it is recommended to type manually echo $ICP_ROOT to verify that the script has set the variables correctly.

Caution: Please make sure that the current date has been set, otherwise the make process goes into an endless loop.

1. The following commands build the EP80579 IP Telephony software, EP80579 security software and EP80579 embedded software drivers:

```
cd $ICP_ROOT
make clean
make
make install
```

Note: A number of warnings will be displayed. These warnings can be ignored. See the IXA000206755 known issue description in the Release Notes for more information.

Note: You may see error output here indicating that icp_debug is already installed. Please ignore the error. The reason for the error is that the make install command invokes the installation script files, install_security.sh and install_voice.sh. These installation script files attempt to install icp_debug. The error is seen when the scripts attempt to install icp_debug after it is already installed.

2. The OCF Shim can be built optionally after building the Acceleration kernel modules using the following command:

```
export ICP_OCF_SRC_DIR=$KERNEL_SOURCE_ROOT/crypto/ocf
make ocf
```

Table 3. Output Files Created in the $ICP_BUILD_OUTPUT Directory

File Name	Description
*.ko	Kernel modules
*.bin	Acceleration Services Unit (ASU) microcode files
icp_asd.conf	Installed in /etc and is the configuration file used by the Acceleration System Driver (ASD)
icp_ctl	User space utility that downloads the /etc/icp_asd.conf file to ASD during initialization
debugmgr	Proprietary debug command to get software version information and data dump
install_security.sh	Installation script for security software

Table 3. Output Files Created in the $ICP_BUILD_OUTPUT Directory

File Name	Description
install_voice.sh	Installation script for IP telephony software
voice_service	Shell script used to Start and Stop IP telephony components (except sRTP)
qat_service	Shell script used to Start and Stop kernel modules

Accelerated (IP Telephony):

- analog_fxo_fxs_driver.ko
- hssvoice_driver.ko
- tdm_infra.ko
- t1e1_framer_infra_driver.ko
- t1e1_framer_driver.a
- tdm_data_driver.ko
- tdm_setup_driver.ko
- srtp_acc_driver.ko

4.4 Optionally Installing Accelerated libSrtp

Note: If you are planning to integrate with the Asterisk HSS channel driver, you may want to skip this section. The Asterisk HSS channel driver has not been tested with the libSrtp Library.

The following is an example of the installation procedure for the modified LibSrtp Library:

1. Download the LibSrtp Library, Version 1.4.2 from the source forge web site (SourceForge.net): http://srtp.sourceforge.net/srtp-1.4.2.tgz
2. Extract the LibSrtp source code to a location chosen by the user. This location can be independent of the EP80579 integrated processor software release package. The extracted folder and its location shall be known as LIBSRTP_FOLDER for the purpose of this example.

```
tar -xvf srtp-1.4.2.tar
export LIBSRTP_FOLDER=$PWD
```

Note: The LibSRTP_V1.4.2_QAT_Support.patch file is provided to change the source files of the LibsRTP library. The patch performs all the necessary code changes to enable LibsRTP for use with the Intel® EP80579 Software for IP Telephony Applications on Intel® QuickAssist Technology. The library can be rebuilt as described in the libsRTP readme.

3. Apply the patch to LIBSRTP_FOLDER/srtp:

```
cd $ICP_ROOT/OpenSourcePatches/
patch -Np1 -d $LIBSRTP_FOLDER/srtp < libSRTP_V1.4.2_QAT_Support.patch
```

Note: The Location of the patch file in the EP80579 integrated processor software Linux Package is:
$ICP_ROOT/OpenSourcePatches

4. Build the LibSrtp Library by changing directory to the LIBSRTP_FOLDER/srtp and running the following commands:

```
cd $LIBSRTP_FOLDER/srtp
./configure
make all
```

Note: When “make all” is performed, the user is prompted to execute “make runtest” at the end of the compilation. Please note that this operation fails if the LibSRTP driver has not been loaded first. The driver can be loaded using the ./load_sRTP_driver script. Please refer to Section 4.5.1.1 for instructions on loading the driver.

Details on the changes applied by the libsRTP patch are given in the Intel® EP80579 Software for IP Telephony Applications on Intel® QuickAssist Technology - SRTP Acceleration Driver Application Note.

Note: There is a compile time switch integrated into the LibSrtp Makefile that enables SRTP acceleration by default. To compile with SRTP acceleration disabled, and to restore original LibSrtp behavior, use the make command with the “SRTP_ACC=OFF” argument:

```
make SRTP_ACC=OFF
```

4.5 Installation of Build Output

4.5.1 IP Telephony

4.5.1.1 Overview of Scripts

Note: The scripts described in this section are automatically called by the software depending on the context as described below. These scripts are **not** intended to be manually run by the user.

The following scripts are included in this release:

- install_voice.sh
- voice_service
- load_sRTP_driver

Commonly used scripts are automatically copied to the BUILD_OUTPUT directory. If the accelerated libSRTP library is being used, please copy the load_sRTP_driver script file from the $ICP_ROOT/Install directory to the $ICP_BUILD_OUTPUT.

install_voice.sh

The install_voice.sh script is called when the top-level make install command is issued, see Section 4.3, “Build Using Top Level Make” on page 33. The script installs and starts all the necessary components for an IP telephony system with the exception of sRTP. sRTP requires Intel® EP80579 integrated processor Security Application kernel modules that are **not** installed from the install_voice.sh script. The installation is persistent. The install_voice.sh script can be used to install and uninstall IP Telephony drivers.

The command formats for executing the install_voice.sh script are as follows:

- For installation, use the command format:

  ```
  ./install_voice.sh install <path to build output directory>
  ```

- For later uninstall, use the command format:

```
./install_voice.sh uninstall <path to build output directory>
```

In both cases, <path to build output directory> is a required parameter.

So, the command to run the script to install the EP80579 IP Telephony software from the $ICP_BUILD_OUTPUT directory is:

```
./install_voice.sh install $ICP_BUILD_OUTPUT
```

The command to run the script to uninstall the EP80579 IP Telephony software is:

```
./install_voice.sh uninstall $ICP_BUILD_OUTPUT
```

voice_service

The voice_service script is used to load or unload all IP telephony components with the exception of sRTP. To load the voice components:

```
./voice_service start
```

To unload the voice components:

```
./voice_service stop
```

Note: It is not necessary to run the voice_service script after the ./install_voice.sh script is run or after each reboot.

load_sRTP_driver

The load_sRTP_driver script is used to load the sRTP driver.

```
./load_sRTP_driver
```

Note: The load_sRTP_driver script must be called after each system boot because it is NOT persistent.

To run any of the scripts, the general procedure is as follows:

1. Ensure that the $ICP_BUILD_OUTPUT environment variable is set and points to the location of the kernel objects to be loaded.
2. Change directory to the $ICP_BUILD_OUTPUT directory.
3. Ensure that the scripts are executable. If they are not, change the permissions on the scripts using the chmod command.

```
./<script_name>
```

The module should be loaded from the $ICP_BUILD_OUTPUT path. The load path is printed when the script runs.

4.5.1.2 Installing and Configuring IP Telephony Drivers

Note: It is a prerequisite that all drivers are built (see Section 4.3, “Build Using Top Level Make” on page 33) before performing the instructions in this section.

IP Telephony drivers can be installed as follows:

1. Run the scripts for installing and configuring drivers:

Note: If make install was executed (see Section 4.3, "Build Using Top Level Make" on page 33), it is not necessary to issue the "./voice_service start" and "./qat_service start" commands following.

```
cd $ICP_BUILD_OUTPUT

./voice_service start
./qat_service start

./load_sRTP_driver
```

Notes: a. If the loading of the icp_asd.ko kernel module fails, check the BIOS NCDRAM and CDRAM size settings and the mem argument in the /boot/grub/grub.conf file of the rebuilt kernel. Also, ensure that you have booted the correct kernel using the command: uname -r. Please refer to Note: for details on the BIOS NCDRAM and CDRAM size settings.

b. The qat_service script must be executed before running the load_sRTP_driver script, since the load_sRTP_driver script is dependent on the Lookaside Crypto (LAC) kernel object files being loaded first.

c. The scripts include instructions for removing the respective driver. If the driver was not loaded, an error is reported and the error messages can be safely ignored.

d. In the event of a permission denial during execution of the above scripts, use the "chmod 777" command, followed by the execution command. For example:

```
chmod 777 voice_service
./voice_service
```

2. To confirm that all the expected drivers have been loaded, use the following command:

   ```
   lsmod
   ```

   At this point, all the expected drivers will be loaded correctly.
   By issuing the lsmod | less command, you can check that the expected drivers have been loaded:
   1) srtp_acc_driver
   2) hssvoice_driver
   3) analog_fxo_fxs_driver
   4) tdm_setup_driver
   5) tdm_data_driver
   6) tdm_infra
   7) t1e1_framer_infra_driver
   and other icp kernel modules as well.

   Driver installation and configuration has been completed. You are now ready to load customer-specific application software that uses these drivers.

   If you subsequently need to unload the voice-only drivers, please refer to the instructions in Section 10.2, "Unloading IP Telephony Drivers" on page 57.

One sample software application is provided for reference. See Chapter 11.0, "Using the HSS Channel Driver for Asterisk". This sample application can be used to verify that the EP80579 IP Telephony software package has been successfully installed.

5.0 Runtime Configuration

A user working in an X-Windows environment may want to execute the following command to disable X-Windows and work in a command line environment:

```
init 3
```

Note: The user can switch amongst six screens by entering the key sequence <ALT><F1| F2| F3| F4| F5| F6>. The “init 5” command takes the user back into X-Windows mode.

5.1 Installing and Configuring Telephony Modules

Refer to Section 4.5.1.2, “Installing and Configuring IP Telephony Drivers” on page 36.

After completing these steps, users can begin to develop their IP Telephony applications. See Chapter 11.0, “Using the HSS Channel Driver for Asterisk” for more information.

Please note that an open source application is not supported by Intel.

5.2 Loading OCF and OCF Shim (Optional for IP Telephony Applications)

Execute the following commands to install OCF for use with the crypto accelerators on the Intel® EP80579 Integrated Processor with Intel® QuickAssist Technology:

```
modprobe ocf
modprobe cryptodev

cd $ICP_BUILD_OUTPUT
modprobe icp_ocf
```

At this point, the user should be able to see the following processes:

- ocf-random (OCF Thread that fills Entropy Pool in Linux)
- ocfLacDregWork (Thread used by OCF Shim for session deregistration)

by executing the following command:

```
ps -ef | grep “ocf”
```

Note: Other processes with names starting with ocfLac may also be displayed.

5.3 Using the Intel® QuickAssist Technology Cryptographic API

At this point, the user can configure and use a Network crypto application such as VPN.

Appendix A, “Build and Install Openswan” describes how to build and install the Openswan IPsec network stack component on the development board.

Appendix B, “Configuring Sample VPN Application” provides instructions for configuring a VPN application using the development board as one of the VPN gateways. The VPN application has been configured and validated with another IA computer running the Red Hat* Enterprise Linux* 5.0 OS and Openswan.

6.0 Building Security and Embedded Components Individually

This chapter describes how to build EP80579 security software and EP80579 embedded software drivers individually.

6.1 Building Components Individually Using Top-Level Make

Note: In the current release, there is no individual build option for IP Telephony components.

The EP80579 embedded software drivers can be built separately using the following command:

```
cd $ICP_ROOT
make Embed_clean
make Embed
make Embed_install
```

The Acceleration kernel modules can be built separately using the following command:

```
cd $ICP_ROOT
make Accel_clean
make Accel
make Accel_install
```

The corresponding uninstall commands are:

```
make Embed_uninstall
make Accel_uninstall
```

7.0 Building, Installing and Loading Individual EP80579 Embedded Software Drivers

Before building and loading EP80579 embedded software drivers drivers, define the following environment variable:

```
export ICP_ROOT=/EP805XX_release
```

Note: As per the instruction in the previous chapters, the kernel source are untarred at KERNEL_SOURCE_ROOT and other sources are untarred at ICP_ROOT.

7.1 Controller Area Network (CAN) Driver

7.1.1 Linux Compilation Instructions

All source files for the Linux release of the Controller Area Network (CAN) driver are located in the following directory within the Linux compatible EP80579 security software release:

```
$ICP_ROOT/Embedded/src/CAN
```

Compilation of the Linux CAN driver separately from the rest of the software package is possible. Change to the $ICP_ROOT/Embedded/src/CAN directory and execute the following commands:

```
make clean
make
```

The CAN driver compiles and the resulting can.ko file is placed in the $ICP_ROOT/Embedded/src/CAN/build/linux_2.6/kernel_space directory.

7.1.2 Linux Module Load/Unload Instructions

To load the Linux CAN driver, execute the following command from the ./build/ linux_2.6/kernel_space/ directory:

```
insmod can.ko
```

Note: The driver can also be installed for persistence using the “make install” command from the /Embedded/src/CAN directory.

To unload the Linux CAN driver, execute the following command:

```
rmmod can.ko
```

The lsmod command may be used to confirm if a module has been loaded or unloaded:

```
lsmod | grep can
```

The output of the above command lists modules loaded in the system containing “can” as part of their name.

7.2 Enhanced Direct Memory Access (EDMA) Driver

7.2.1 Linux Compilation Instructions

All source files for the Linux release of the Enhanced Direct Memory Access (EDMA) driver are located in the following directory within the Linux compatible EP80579 security software release:

```
$ICP_ROOT/Embedded/src/EDMA
```

Compilation of the Linux EDMA driver separately from the rest of the software package is possible. Change to the $ICP_ROOT/Embedded/src/EDMA directory and execute the following commands:

```
make clean
make
```

The EDMA driver compiles and the resulting edma.ko file is placed in the $ICP_ROOT/Embedded/src/EDMA/build/linux_2.6/kernel_space directory.

7.2.2 Linux Module Load/Unload Instructions

To load the Linux EDMA driver, execute the following command from the directory where the compiled executable resides:

```
insmod dma.ko
```

Note: The driver can also be installed for persistence using the “make install” command from the /Embedded/src/EDMA directory.

To unload the Linux EDMA driver, execute the following command:

```
rmmod dma.ko
```

The lsmod command may be used to confirm if a module has been loaded or unloaded:

```
lsmod | grep dma
```

The output of the above command lists modules loaded in the system containing “dma” as part of their name.

7.2.3 Runtime Configuration of EDMA

Runtime configuration of the Enhanced Direct Memory Access driver is performed from a client driver communicating through the published API set as described in the Intel® EP80579 Software Drivers for Embedded Applications Programmer’s Guide and API Reference Manual. Intel leaves the design and development of a client driver to the customer since it is implementation specific.

7.3 Watchdog Timer (WDT) Driver

7.3.1 Linux Compilation Instructions

All source files for the Linux release of the Watchdog Timer (WDT) driver are located in the following directory within the Linux compatible EP80579 security software release:

```
$ICP_ROOT/Embedded/src/WDT
```

Compilation of the Linux WDT driver separately from the rest of the software package is possible. Change to the $ICP_ROOT/Embedded/src/WDT directory and execute the following commands:

```
make clean
make
```

The Watchdog Timer driver compiles and the resulting wdt.ko file is placed in the $ICP_ROOT/Embedded/src/WDT/build/linux_2.6/kernel_space directory.

7.3.2 Linux Module Load/Unload Instructions

To load the Linux Watchdog Timer driver, execute the following command from the directory where the compiled executable resides ($ICP_ROOT/Embedded/src/WDT/build/linux_2.6/kernel_space):

```
insmod wdt.ko
```

To unload the Linux Watchdog Timer driver, execute the following command:

```
rmmod wdt.ko
```

Note: The driver can also be installed for persistence using the “make install” command from the /Embedded/src/WDT directory.

The lsmod command may be used to confirm if a module has been loaded or unloaded:

```
lsmod | grep wdt
```

The output of the above command lists modules loaded in the system containing “wdt” as part of their name.

7.3.3 Runtime Configuration of WDT

Runtime configuration of the Watchdog Timer driver is performed from a client driver communicating through the published API set as described in the Intel® EP80579 Software Drivers for Embedded Applications Programmer’s Guide and API Reference Manual. Intel leaves the design and development of a client driver to the customer since it is implementation specific.

7.4 General Purpose I/O (GPIO) Driver

7.4.1 Linux Compilation Instructions

All source files for the Linux release of the General Purpose I/O (GPIO) driver are located in the following directory within the Linux compatible EP80579 security software release:

```
$ICP_ROOT/Embedded/src/GPIO
```

Compilation of the Linux GPIO driver separately from the rest of the software package is possible. Change to the $ICP_ROOT/Embedded/src/GPIO directory and execute the following commands:

```
make clean
make
```

The GPIO driver compiles and the resulting gpio.ko file is placed in the $ICP_ROOT/Embedded/src/GPIO/build/linux_2.6/kernel_space directory.

7.4.2 Linux Module Load/Unload Instructions

To load the Linux General Purpose I/O driver, execute the following command from the directory where the compiled executable resides:

```
insmod gpio.ko
```

Note: The driver can also be installed for persistence using the “make install” command from the /Embedded/src/GPIO directory.

To unload the Linux General Purpose I/O driver, execute the following command:

```
rmmod gpio.ko
```

The lsmod command may be used to confirm if a module has been loaded or unloaded:

```
lsmod | grep gpio
```

The output of the above command lists modules loaded in the system containing “gpio” as part of their name.

7.4.2.1 Runtime Configuration of GPIO

Runtime configuration of the General Purpose I/O driver is performed from a client driver communicating through the published API set as described in the Intel® EP80579 Software Drivers for Embedded Applications Programmer’s Guide and API Reference Manual. Intel leaves the design and development of a client driver to the customer since it is implementation specific.

7.5 IEEE 1588 Hardware Assist Driver

7.5.1 Linux Compilation Instructions

All source files for the Linux release of the IEEE 1588 Hardware Assist (1588) driver are located in the following directory within the Linux compatible EP80579 security software release:

```
$ICP_ROOT/Embedded/src/1588
```

Compilation of the Linux IEEE 1588 driver separately from the rest of the software package is possible. Change to the /Embedded/src/1588 directory and execute the following commands:

```
make clean
make
```

The IEEE 1588 Hardware Assist driver compiles and the resulting timesync.ko file is placed in the /Embedded/src/1588/build/linux_2.6/kernel_space directory.

7.5.2 Linux Module Load/Unload Instructions

To load the Linux IEEE 1588 Hardware Assist driver, execute the following command from the directory where the compiled executable resides:

```
insmod timesync.ko
```

To unload the Linux IEEE 1588 Hardware Assist driver, execute the following command:

```
rmmod timesync.ko
```

Note: The driver can also be installed for persistence using the “make install” command from the /Embedded/src/1588 directory.

The lsmod command may be used to confirm if a module has been loaded or unloaded:

```
lsmod | grep timesync
```

The output of the above command lists modules loaded in the system containing “timesync” as part of their name.

7.5.3 Runtime Configuration of IEEE 1588

Runtime configuration of the IEEE 1588 Hardware Assist driver is performed from a client driver communicating through the published API set as described in the Intel® EP80579 Software Drivers for Embedded Applications Programmer’s Guide and API Reference Manual. Intel leaves the design and development of a client driver to the customer since it is implementation specific.

7.6 Global Configuration Unit and Gigabit Ethernet Drivers

Two drivers are required for enabling network connectivity on the Gigabit Ethernet controllers in the EP80579: the Global Configuration Unit (GCU) driver and the Gigabit Ethernet (GbE) driver. The GCU driver controls the MAC and administrative activities. The GbE driver controls the network connectivity. The GbE driver is dependent on the GCU driver.

Note: The Global Configuration Unit driver must be installed prior to installation of the Gigabit Ethernet driver.

7.6.1 Linux Compilation Instructions

All source files for the Linux release of the Global Configuration Unit (GCU) driver and the Gigabit Ethernet (GbE) driver are located in the following directory within the Linux compatible EP80579 security software release:

```
$ICP_ROOT/Embedded/src/GbE
```

Compilation of both the GCU and GbE drivers separately from the rest of the software package is possible. Enter the $ICP_ROOT/Embedded/src/GbE directory and execute the following commands:

```
make clean
make
```

The GCU and GbE drivers compile and the resulting gcu.ko and iegbe.ko files are placed in the $ICP_ROOT/Embedded/src/GbE/build/linux_2.6/kernel_space directory.

7.6.2 Linux Module Load/Unload Instructions

To load the Linux Global Configuration Unit driver, execute the following command from the directory where the compiled executable resides ($ICP_ROOT/Embedded/GbE):

```
insmod gcu.ko
insmod iegbe.ko
```

To unload the Linux Global Configuration Unit driver, execute the following commands:

```
rmmod gcu.ko
rmmod iegbe.ko
```

The lsmod command may be used to confirm if a module has been loaded or unloaded:

```
lsmod | grep gcu
lsmod | grep iegbe
```

The output of the above commands list modules loaded in the system containing “gcu” and “iegbe” as part of their name.

7.6.3 Runtime Configuration of GCU and Gigabit Ethernet

Runtime configuration of network support is through traditional ifconfig commands, detailed in man pages installed on the system.

Note: In Linux, the Ethernet ports TT, SS and UU in Figure 4 on page 17 appears as eth0, eth1 and eth2 respectively.

7.6.4 Alternative PHY Compilation

This section covers compilation instructions for the Gigabit Ethernet driver that has been modified to support the National Semiconductor* DP83848I PHY.

7.6.4.1 Linux Compilation Instructions

All source files for the Gigabit Ethernet Driver patch are located in the following directory:

```
/Embedded/src/AlternativePHY/DP83848I
```

Enter this directory using the following command:

```
cd /Embedded/src/AlternativePHY/DP83848I
```

Build the GbE and GCU drivers within the Gigabit Ethernet driver patch package using the following command:

```
make clean
make
```

To install the drivers for persistence on future boot, execute the following command from the /Embedded/src/AlternativePHY/DP83848I directory:

```
make install
```

Note: If the top-level make file described in Section 4.3, “Build Using Top Level Make” on page 33 is executed after the make install described in this section, the alternative Gigabit Ethernet driver will be replaced with the standard Gigabit Ethernet driver and will be loaded on subsequent reboots.

7.6.4.2 Linux Module Load/Unload Instructions

To load the alternative Global Configuration Unit driver and the Gigabit Ethernet driver manually, execute the following commands from the directory where the compiled executable resides (/EP805XX_release/Embedded/GbE/build/linux_2.6/kernel_space):

```
insmod gcu.ko
insmod iegbe.ko
```

To unload the Linux Global Configuration Unit driver and Gigabit Ethernet driver execute the following commands:

```
rmmod gcu.ko
rmmod iegbe.ko
```

The lsmod command may be used to confirm if a module has been loaded or unloaded:

```
lsmod | grep gcu
lsmod | grep iegbe
```

The output of these lsmod command lists modules loaded in the system containing "gcu" and "iegbe" as part of their name.

7.7 System Management Bus (SMBus) Driver

7.7.1 Linux Compilation Instructions

All source files for the SMBus Linux support are available in the Linux kernel source. Additionally, the lm_sensors rpm must be installed to work with additional SMBus hardware on the system board. The lm_sensors rpm can be acquired through the Red Hat Network, and the source RPM can be found on Disc 3 of the Red Hat Enterprise Linux 5.0 client source installation CDs. Downloading and installing the RPM rather than the source RPM won't require a build of the source first.

Note: Before installing lm_sensors rpm, you can check if it has already been installed using the `rpm -q lm_sensors` command.

Download lm_sensors-2.10.0-3.1.i386 RPM package from the Red Hat Network and install the rpm with the following command:

```
rpm -ivh lm_sensors-2.10.0-3.1.i386
```

Previously, in the installation process (Section 3.6, "Patching the Kernel for PCI Device Recognition" on page 27), the patch file Intel_EP80579_RHEL5.patch was applied. This patch file applied updates to the following files:

```
drivers/i2c/busses/Kconfig
drivers/i2c/busses/i2c-i801.c /*chipset hardware access driver*/
```

Two additional SMBus drivers, which are available within the Linux kernel source, need to be installed prior to use of the i2c-i801 driver.

```
i2c-core
i2c-dev
```

These drivers are available within the Linux kernel source:

```
$KERNEL_SOURCE_ROOT/drivers/i2c/i2c-core.ko
$KERNEL_SOURCE_ROOT/drivers/i2c/i2c-dev.ko
```

It is likely that these drivers will be installed during installation of the operating system and kernel. To determine if they are installed perform an lsmod similar to the examples above.

7.7.2 Linux Module Load/Unload Instructions

If the additional SMBus driver files are not installed, install them with either the modprobe or insmod commands:

```
modprobe i2c-core
mobprobe i2c-dev
```

or

```
insmod /lib/modules/`uname -r`/kernel/drivers/i2c/i2c-core.ko
insmod /lib/modules/`uname -r`/kernel/drivers/i2c/i2c-dev.ko
```

Installing the i2c-i801 driver is accomplished using the following command:

```
modprobe i2c-i801
```

or

```
insmod /lib/modules/`uname -r`/kernel/drivers/i2c/busses/i2c-i801.ko
```

To unload the Linux SMBus i2c-i801 driver execute the following command:

```
rmmod i2c-i801.ko
```

The lsmod command may be used to confirm if a module has been loaded or unloaded:

`lsmod | grep i2c_i801` (underscore and not dash)

The output of the above command lists modules loaded in the system containing "`i2c_i801`" as part of their name.

7.7.3 Runtime Configuration of SMBus

Runtime configuration of the SMBus driver is performed from a client driver communicating through the published API set as described in the Intel® EP80579 Software Drivers for Embedded Applications Programmer's Guide and API Reference Manual. Examples of already developed client drivers in the kernel source tree are i2cdetect or i2cdump. Intel leaves the design and development of a client driver to the customer since it is implementation specific.

8.0 Building and Using Crypto and Debug Tools

Debugmgr is a user space command and debugmgmt.ko is the corresponding kernel module that is built when building the EP80579 security software kernel modules. Cryptotest is an open source user space command that invokes crypto operations using OCF.

8.1 Building Crypto Tools

Build the Crypto tools as follows:

1. Save the crypto-tools-20070727.tar.gz file from the following location

   http://heanet.dl.sourceforge.net/sourceforge/ocf-linux/crypto-tools-20070727.tar.gz

   to the $ICP_ROOT directory and change to that directory:

   ```
   cd $ICP_ROOT
   ```

2. Extract the crypto-tools package using the following command:

   ```
   tar -xvzf crypto-tools-20070727.tar.gz
   ```

3. Change directory to the newly extracted crypto-tools directory:

   ```
   cd crypto-tools
   ```

4. Due to a BSD system call in crypto-tools that is not present in Linux, it is necessary to edit any “strlcpy” calls to be “strncpy” calls in the cryptotest.c and cryptokeytest.c files. The following commands can be used for this purpose:

   ```
   sed -i s/strlcpy/strncpy/g cryptotest.c
   sed -i s/strlcpy/strncpy/g cryptokeytest.c
   ```

5. Compile the code files:

   ```
   make
   ```

Note: The cryptokeytest may not compile successfully. Even if Cryptokeytest compiles, it will not work because it differs from implementation of the feature in the EP80579 integrated processor. The issue with cryptokeytest does not impact cryptotest.

8.2 Using the Crypto Tools

The cryptotest command can be found in the $ICP_ROOT/crypto_tools directory.

Note: The user must load the OCF stack for this command to work.

The following is the cryptotest command description.

Command:

```
cryptotest [-csbv] [-a algorithm] [count] [size ...]
```

Description:

Runs an encrypt+decrypt or mac operation on a buffer of “size” bytes for a “count” number of iterations. A random key and IV are used.

Options:

- -c check the results
- -d dev pin work on device dev
- -v verbose mode
- -b mark operations for batching
- -p profile kernel crypto operations (must be root)
- -t n fork n threads and run test concurrently

Supported Algorithms:

- -3des Triple DES in CBC mode
- -sha1 SHA-1
- -sha1_hmac HMAC SHA-1

Example:

```
cryptotest -a 3des 100 8192
```

Note: The 3des option requires the test size to be a multiple of 8; it fails silently if not.

8.2.1 Using Debug Tools

Load the debug manager using the following command:

```
cd /EP805XX_release
export ICP_ROOT=$PWD
insmod $ICP_ROOT/StagingArea/icp_debugmgmt.ko
```

Create a character device /dev/mil_driver using the following steps:

1. Find the device major number associated with the character device /dev/mil_dirver using the following command:

   ```
   cat /proc/devices | grep mil_driver
   ```

   The following is a sample output from the above command:

   ```
   249 /dev/mil_driver
   ```

2. Use the device major number displayed by the above command to create the character device using the following command:

   ```
   mknod /dev/mil_driver c <device major number> 0
   ```

   If we use the sample output from step 1, then the mknod command would be:

   ```
   mknod /dev/mil_driver c 249 0
   ```

Then, to use the debugmgr, the following command is executed from where the release objects are stored. The following options are currently supported:

```
debugmgr
debugmgr DebugEnable
debugmgr DebugDisable
debugmgr VersionDumpAll
debugmgr setHC <time-out in miliseconds> Range[100-5000]
debugmgr SystemHealthCheck
debugmgr DataDump
```

Note: The debugmgr must be enabled using the option ‘DebugEnable’ before it can display information and disabled using the option ‘DebugDisable’ at the end of it.

9.0 Pre-boot (BIOS) Firmware

The pre-boot firmware is executed when the system is powered up or reset. It initializes and configures system memory, devices and interfaces.

The Pre-boot Firmware is based on the AMI Aptio* 4.5 core and compliant to Extensible Firmware Interface (EFI) v1.1. The firmware is stored in the Firmware Hub (FWH) or Serial Peripheral Interface (SPI) Flash; the FWH or SPI Flash can be updated using a flash utility tool that is provided.

The pre-boot firmware setup menu can be used to view and modify the system settings for the development board. The setup menu is accessed by pressing the <Del> key during pre-boot firmware boot up (before the operating system begins). The setup menu bar is shown in Table 4.

9.1 Pre-boot Firmware Setup Menu

Table 4 shows the pre-boot firmware setup main menu and provides a brief description of each menu option. Table 5 shows the action keys that can be used when navigating and selecting options from pre-boot firmware menus.

Table 4. Pre-boot Firmware Setup Main Menu

Main	Advanced	Chipset	Security	Boot	Exit
Displays processor and memory configuration Setup for CMOS system date and time	Configures advanced features and settings	Configures different major components	Setup passwords and security features	Selects boot options and configurations	Saves or discards changes to setup program options

Table 5. Pre-boot Firmware Setup Program Action Keys

Function Key	Description
< or >	Moves cursor left or right in the main menu
^ or v	Moves cursor up or down to select sub-menu items
Enter	Executes command or selects the submenu
F7	Discard changes
F8	Load the fail-safe default
F9	Load the optimal default configuration value for the current menu
F10	Save the current configuration and exit the setup menu
ESC	Exit the setup menu

9.1.1 Serial Console Redirection

The Pre-boot Firmware supports redirection of both video and keyboard via a serial port. When console redirection is enabled, the remote console terminal sends keystrokes to the development board Pre-boot Firmware and the Preboot Firmware redirects the video to the console terminal.

As an option, the development board can be operated without keyboard or video and can run entirely via the remote serial console. This includes accessing the Pre-boot Firmware setup menu.

Console redirection ends when operating system boot up begins. After boot up begins, the operating system is responsible for continuing the redirection.

Note: Pre-boot Firmware console redirection is text only. Graphical data, such as logos, are not redirected.

Table 6 shows the default settings of the serial console redirection.

Table 6. Serial Console Redirection Default Settings

Parameter	Default
Port Number	COM 1
Baud Rate	115200
Data Bits	8
Parity	None
Stop bits	1
Flow Control	None

9.1.2 Changing the Boot Device

Use the following procedure to change the boot device:

1. Press the <Del> key during Power On Self Test (POST) to enter the pre-boot firmware setup menu.
2. Use the arrow keys to navigate to the <BOOT> menu.
3. Move the cursor to <Boot Device Priority>.
4. Select the desired booting sequence list.

Note: Follow the instructions on the right side of the pre-boot firmware screen to navigate and change pre-boot firmware settings.

9.1.3 Maximum Memory Speed Setup

The maximum memory speed supported on the development board can be selected using the Maximum Memory Speed Setup option available in the BIOS Setup Menu on the Chipset tab.

Enter the BIOS Setup Menu and select the Chipset tab. Select the North Bridge sub tab. Navigate down to the bottom option, titled Max Memory Speed Support, and select this option using the Enter button. A selection box appears providing the following options:

- 400 MHz
- 533 MHz
- 667 MHz

- 800 MHz

The default setting in the BIOS is 400 MHz. If a higher speed memory DIMM is inserted into the development board, the corresponding memory speed must be selected in the BIOS Setup Menu to support the intended speed. Otherwise, the memory is reduced to the default of 400 MHz.

9.1.4 Coherent and Non-Coherent Memory Allocation

The development board supports allocation of memory regions for coherent and non-coherent use. Coherent and non-coherent memory use features are for development boards that use the Intel® EP80579 Integrated Processor with Intel® QuickAssist Technology. EP80579 integrated processors without Intel® QuickAssist Technology do not make use of the memory set aside for these features.

Enter the BIOS Setup Menu and select the Chipset tab. Select North bridge, and navigate down towards the bottom to the Coherent Mem Size option and press Enter to select this option. A dialog box is displayed prompting the user to enter a value. Type the numerical value “2000”, and press Enter. Navigate to the next option, Non-Coherent Mem Size, and press Enter to select this option. A dialog box is displayed prompting the user to enter a value. Again, type the numerical value zero, “2000”, and press enter. These selections override the BIOS default settings and allocates memory regions for these two features.

Note: The proposed values of 2000 for Coherent and Non-Coherent Mem Size is tentative for GA1.0 only. A table with recommended values for Coherent and Non-Coherent Mem Size for various memory configurations is planned for future releases.

Note: This software package requires the pre-boot firmware (BIOS) for your hardware to allocate the values for each region called out in Table 7. For more information on these regions, refer to the Intel® EP80579 Integrated Processor Product Line Datasheet, Section 3.0.

Table 7. Memory Allocation Settings

Datasheet name	Software name	Region Size
IA/ASU Shared (Coherent)	CDRAM	32MB
IA/ASU Shared (AIOC-Direct)	NCDRAM	32MB

9.1.5 Legacy and AHCI SATA Mode

The development board supports hard drives in legacy SATA mode and in Advanced Host Controller Interface (AHCI) mode. AHCI mode provides advanced capabilities and improved performance, provided the hard drive supports the following features:

- Hot Plug
- Native Command Queuing
- Speeds up to 3 Gb/s

Refer to the Serial ATA Organization web site for more information:

http://www.serialata.org/

The development board pre-boot firmware (BIOS) can be configured in either Legacy or AHCI mode as desired. The BIOS defaults to Legacy mode because not all hard drives support AHCI. To toggle the BIOS to either Legacy or AHCI mode, proceed as follows:

1. Press the <Del> key during POST to enter the pre-boot firmware setup menu.
2. Use the arrow keys to navigate to the Advanced menu.
3. Use the arrow keys to navigate to the IDE Configuration option.
4. Select the IDE Configuration option.
5. Use the arrow keys to navigate to the SATA Mode option.
6. Press the Enter key. A SATA Mode popup window appears.
7. Select either Legacy or AHCI as desired. Do not use Native as a selection.
8. Press F4 to save.
9. Choose Yes. The system continues the boot process.

9.2 Pre-boot Firmware Image Reflashing Instructions

One method is available for updating the pre-boot firmware flash images located on the development board FWH or SPI Flash:

- AFUEFI Flash Recovery

It is possible that updated pre-boot firmware images will become available from Intel for the Intel® EP80579 Integrated Processor with Intel® QuickAssist Technology Development Board. The latest pre-boot firmware image is available from Intel's public web site, http://developer.intel.com located with all other collateral related to the EP80579 integrated processor.

If the pre-boot firmware image should become corrupted on the board, also utilize these instructions to reflash the image.

9.2.1 Aptio Flash Update Utility (AFUEFI)

Use the following instructions to update the development board pre-boot firmware image using a USB flash drive and the Aptio Flash Update Utility from AMI.

Necessary Hardware:

- development board
- Socketed Firmware Hub or SPI Flash
- USB flash drive

Necessary Software:

- development board pre-boot firmware Image
- Aptio Flash Update Utility from AMI - AFUEFI

Steps to reflash the image:

1. Execute the AFUEFI utility onto the USB flash drive.
2. Load the pre-boot firmware image onto the USB flash drive.
3. Change the boot setting in the BIOS Setup Selection to boot from the EFI shell. Boot the development board to the EFI shell.
4. Insert the USB flash drive into the USB port.

5. Once the USB flash drive is recognized on the system (activity can be seen on the USB flash drive LED if present), several commands are available as follows:
   — Type “map -r” to list all devices available.
   — Type “fs0:” to enter USB device.
   — Type “ls” to list all files.
6. Once the “fs0:” command has been initiated, execute the AFUEFI utility. Enter:

   ```
   AFUEFI <pre-boot firmware image name> /X /P /B /N
   ```

   where the <pre-boot firmware image name> will be TRXTA055.ROM or similar
7. Reboot the development board when reflashing has completed.
8. Confirm that the image has been updated to the reflashed image by checking the Flash Image identity in the BIOS Setup.

10.0 Uninstalling the Software

Please refer to instructions on loading and unloading individual modules in Section 7.1 through Section 7.7.

Refer to instructions for unloading IP Telephony drivers in Section 10.2.

10.1 Linux Module / Driver Dependencies

Table 8 lists the dependencies for the driver modules or patch within the EP80579 security software package. OS installation is assumed.

Table 8. Linux Module / Driver Dependencies

Module / Driver / Patch	Dependency 1	Dependency 2
EP80579 integrated processor Linux Patch	OS Source	None
OCF	OS Source	None
Openswan	OpenSSL	GMP library
cryptodev	ocf	
icp_ocf	ocf	LAC
icp_asd	icp_crypto	icp_debug
icp_crypto	icp_services	icp_debug
icp_services	icp_hal	icp_debug
icp_debugmgmt	icp_debug	
Controller Area Network (CAN)	None	-
Enhanced DMA (EDMA)	None	-
IEEE 1588 Hardware Assist (1588)	None	-
General Purpose IO (GPIO)	None	-
Gigabit Ethernet (GbE)	GCU	None
Global Configuration Unit (GCU)	None	-
SMBus (i2c-i801)	i2c-core	i2c-dev
Watchdog Timer (WDT)	None	-

Note: For IP Telephony driver dependencies, refer to Section 4.5.1.2.

10.2 Unloading IP Telephony Drivers

All voice and data drivers (except srtp_acc_driver) can be unloaded with the voice service script.

```
cd $ICP_BUILD_OUTPUT
./voice_service stop
```

Note: If security drivers are also installed when the voice components are unloaded an error will be reported as it attempts to remove icp_debug.ko which is also used by the security modules. This error can be ignored.

Alternatively the drivers can be removed using the rmmod command as shown below:

- Unload voice-only drivers as follows:

```
rmmod hssvoice_driver
rmmod analog_fxo_fxs_driver
rmmod t1e1_framer_infra_driver
rmmod tdm_setup_driver
rmmod srtp_acc_driver
rmmod tdm_infra
```

- Unload data-only (HDLC) drivers as follows:

```
rmmod srtp_acc_driver.ko
rmmod tdm_data_driver.ko
rmmod t1e1_framer_infra_driver.ko
rmmod tdm_setup_driver.ko
rmmod tdm_infra.ko
```

11.0 Using the HSS Channel Driver for Asterisk

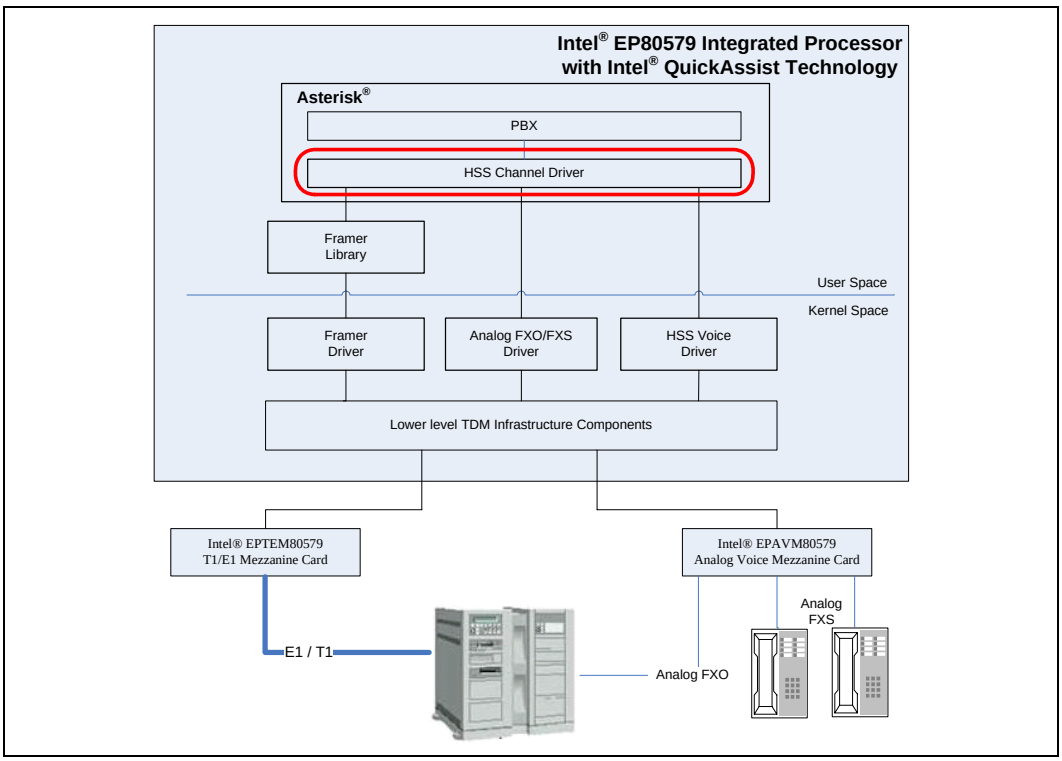
This chapter describes how Asterisk integrates with the TDM Infrastructure of the Intel® EP80579 Integrated Processor with Intel® QuickAssist Technology. It provides an architectural overview along with instructions for installing and building the High Speed Serial (HSS) channel driver for Asterisk. It is assumed that the user has basic knowledge of Asterisk software.

11.1 Architectural Information

In Asterisk, there is an abstraction between PBX functionality and the various technologies. The PBX core is in charge of switching channels, whereas call processing is centered around channel drivers. These drivers handle all the details specific to a technology or a protocol, and interface to Asterisk. The drivers are built as shared objects (.so) and are loaded dynamically into Asterisk as modules.

To integrate Asterisk with the processor's TDM Infrastructure, a channel driver called "chan_hss" was developed. This driver interfaces with the processor's TDM Infrastructure software components (HSS voice driver, analog FXO/FXS driver and framer library) to enable Asterisk to use external hardware for telephony applications. Figure 6 shows the location of this layer in the Asterisk architecture.

Figure 6. TDM Infrastructure with HSS Channel Driver Overview



The features available in the current version of chan_hss driver are:

- Support for up to 12 FXS and three FXO lines
- Support for four digital trunks (four E1s or four T1s)
- Ear & Mouth signaling on digital links (E1 or T1)
- G.711 A-law and u-law, and linear (FXO/FXS only) codecs supported on the connection between Asterisk and POTS phones
- Echo cancellation using OSLEC* (Open Source Line Echo Canceller)
- Call waiting on FXS

There are two mezzanine cards that can be connected to the Intel® EP80579 Integrated Processor with Intel® QuickAssist Technology Development Board:

- Intel® EPAVM80579 Analog Voice Mezzanine Card that supports four FXS lines and one FXO line
- Intel® EPTEM80579 T1/E1 Mezzanine Card that supports four E1/T1 links

Figure 7 shows a layout where two analog voice mezzanine cards and one T1/E1 mezzanine card are connected to the development board. This setup supports in total:

- Eight FXS lines
- Two FXO lines
- One framer which can be configured either for a single or quad digital link

Figure 7. Two analog voice mezzanine cards and one T1/E1 mezzanine card

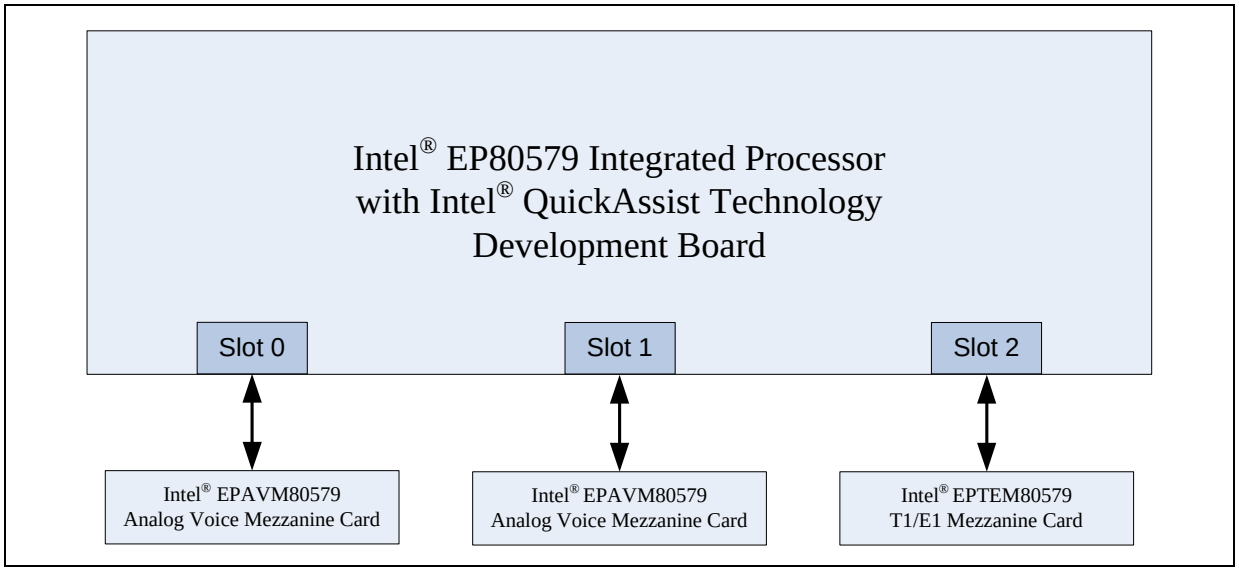
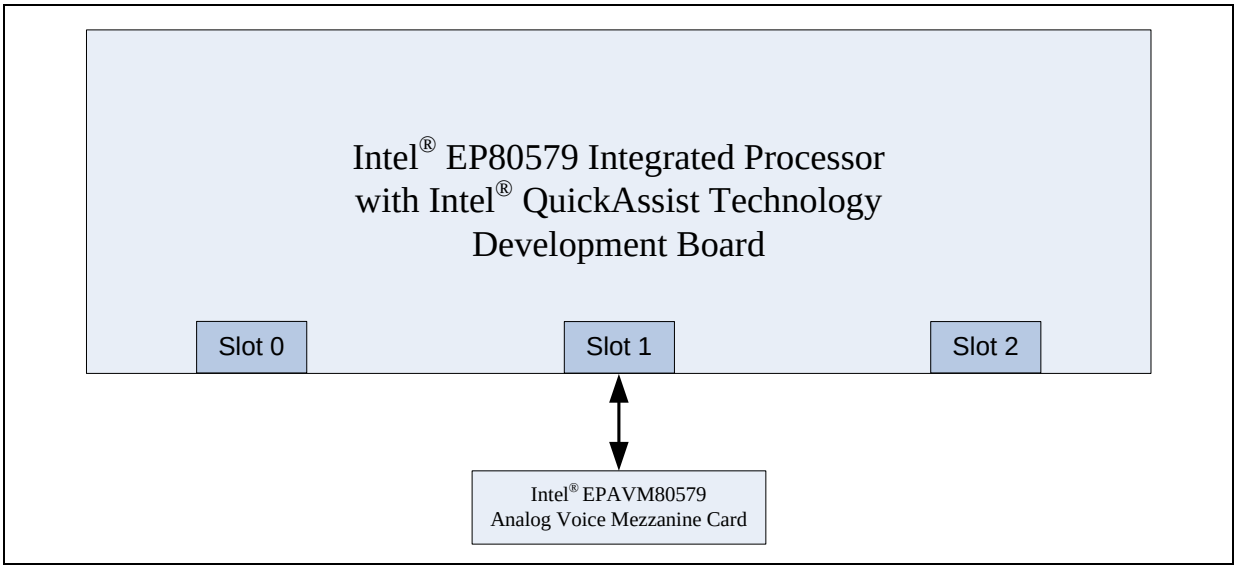


Figure 8 shows a layout where a single analog voice mezzanine card is connected to the development board. This setup supports in total:

- Four FXS lines
- One FXO line

Figure 8. Single analog voice mezzanine card



11.2 Design Considerations

11.2.1 Functional Model

The HSS Channel Driver is responsible for monitoring the lines to detect incoming calls and controlling the phones on outgoing calls. It has interactions with both Asterisk and the processor's TDM Infrastructure components.

On one side, it implements a set of callbacks required by Asterisk to perform actions on the phones (such as ring, hang up and so on). Upon loading, it registers a new channel technology, specifying the codecs capabilities and pointers to these callbacks.

On the other side, the channel driver uses the processor's TDM Infrastructure drivers to control the lines:

- The analog FXO/FXS driver enables monitoring the phones states
- The framer library enables monitoring events coming from the digital lines
- A voice channel is created for each line through the HSS voice driver to exchange voice data

11.2.2 Threading Model

In the HSS Channel Driver, threads are used for different purposes:

- **Analog lines monitoring:**
  One thread is in charge of monitoring the events coming from the FXO/FXS lines. It opens a file descriptor and waits for events coming from the analog FXO/FXS driver. This thread can monitor several analog voice mezzanine cards plugged into any of the three HSS ports.

- **Digital lines monitoring:**
  Another thread monitors the digital lines. As the framer library reports events for each slot separately, it is necessary to start a new instance of this thread for each HSS port configured for a T1/E1 mezzanine card.
- **Extension retrieval:**
  When one of the monitor threads detects that a phone has gone off hook because a user wants to make a call, a new thread is started to detect the dialed digits and build the extension number to be called and passed to Asterisk.
- **Flash hook detection:**
  When the call waiting feature is enabled, flash hook is used to switch between calls. To detect these flash hook events, a new thread is started each time an FXS goes on hook to determine whether the user has ended the call or just sent a flash hook.

The global array containing the interfaces is the only resource that needs to be protected by a mutex since it is accessed by the various threads and the Asterisk callbacks.

11.2.3 Bill of Materials

File	Description
chan_hss.c	source file of the HSS Channel Driver
install.sh	installation script
hss.conf	sample configuration file

11.3 Debug Considerations

When debugging, it is necessary to determine whether the problem comes from the EP80579 IP Telephony software or from Asterisk. To do so, it is possible to use the debug feature to see information from the HSS Channel Driver. The debug information can be turned on from the Asterisk command line by typing:

```
core set debug 1
```

In addition the following command is provided by the HSS Channel Driver to print out the framer lines status from the Asterisk command line:

```
hss framer status
```

For problems related to Asterisk, refer to the documentation available on the wiki page at http://www.voip-info.org/wiki-Asterisk+debugging.

For problems associated with the underlying transmission of voice samples, refer to the TDM Infrastructure documentation.

11.4 Performance Considerations

For maintainability and readability purposes, a decision was taken to store data in a static array. It is possible to improve performance through the use of a different storage structure, such as linked lists, to reduce the time needed to search for a line in the list and to have better memory management, especially when handling a large number of channels.

11.5 Installation Instructions

11.5.1 OS, Kernel, Libraries and Tools Release Requirements

The HSS Channel Driver requires:

- Intel® EP80579 Integrated Processor with Intel® QuickAssist Technology
- Intel® EP80579 Software for IP Telephony Applications on Intel® QuickAssist Technology
- Intel® EP80579 Software for IP Telephony Applications - HSS Channel Driver for Asterisk (see Section 1.2.1 for access information)
- Linux* RHEL 5.0, with the following installed:
  - gcc-c++ (gcc-c++-4.1.1.52.el5.i386.rpm)
  - libstdc++-devel (libstdc++-devel-4.1.1-52.el5.i386.rpm)
  - libtiff-devel (libtiff-devel-3.8.2-7.el5.i386.rpm)
- Kernel 2.6.18
- Asterisk 1.4.13 (http://downloads.digium.com/pub/asterisk/releases/asterisk-1.4.13.tar.gz)
  - The readme file has important instructions for Asterisk build and install. The readme file for Asterisk version 1.4.13 should be downloaded from http://downloads.digium.com/pub/asterisk/releases/README-1.4.13.
- OSLEC 0.1 (http://www.rowetel.com/ucasterisk/downloads/oslec-0.1.tar.gz)

Intel currently supports the version of Asterisk listed above only. Subsequent releases of Asterisk may require updates to the channel driver due to API changes.

11.5.2 Enabling Echo Cancellation

The HSS Channel Driver performs echo cancellation in user space, therefore it does not need the kernel module provided by OSLEC. It just needs the modified version of spandsp* library available in the OSLEC package.

It is therefore necessary to enable the spandsp library to be dynamically linked to applications like Asterisk. To do this, create a file /etc/ld.so.conf.d/libspandsp.conf and in it, put the line "/usr/local/lib" without the quotes. Save the file.

Note: If problems are encountered when building the spandsp library, please see http://www.soft-switch.org/installing-spandsp.html.

Note: As indicated in Section 11.5.1, “OS, Kernel, Libraries and Tools Release Requirements” on page 63, the following rpms need to be installed before issuing the make command:
— gcc-c++ (gcc-c++-4.1.1.52.el5.i386.rpm)
— libstdc++-devel (libstdc++-devel-4.1.1-52.el5.i386.rpm)
— libtiff-devel (libtiff-devel-3.8.2-7.el5.i386.rpm)

Build the spandsp library from OSLEC as follows:

```
tar xvzf oslec-0.1.tar.gz
cd oslec-0.1/spandsp-0.0.3/
./configure
make
make install
```

Run ldconfig to load the new configuration.

11.5.3 Building Asterisk Instructions

Once OSLEC installation is complete, make sure that ICP_ROOT and ICP_BUILD_OUTPUT environment variables are set as described in Chapter 4.0, “Building EP80579 IP Telephony Software on a Target Development Board”, and build Asterisk as usual.

Untar Asterisk by executing the following command:

```
tar -zxvf asterisk-1.4.13.tar.gz
```

Change directory:

```
cd <asterisk-path>
```

Now build Asterisk. For instructions on how to build Asterisk, please refer to the Readme file located in the Asterisk directory.

Execute all of the steps in the Asterisk readme file up to the point of launching the application. The launching of the application should be done after modifications to the configuration files.

Section 11.5.5.1, “Example Configuration Files” on page 66 are the configuration files (both hss.conf and extensions.conf) that need to be created in a Linux* text editor.

The readme file for Asterisk 1.4.13 can also be downloaded from: http://downloads.digium.com/pub/asterisk/releases/README-1.4.13.

11.5.4 Module Install Requirements

It is assumed that the EP80579 IP Telephony software has been installed as described in Chapter 4.0, “Building EP80579 IP Telephony Software on a Target Development Board”.

Download the Intel® EP80579 Software for IP Telephony Applications - HSS Channel Driver for Asterisk package and unzip it:

1. Obtain Telephony.L.1.0.xx_Asterisk_002.tar.gz from the “Tools & Software” tab of the resource center as specified in Section 1.2.1, “Where to Find Current Software and Documentation” on page 7.
2. Copy the file to a directory on target platform, a location such as $ICP_ROOT.
   Using a Flash drive or CDROM, copy the kernel source rpm package to the $ICP_ROOT directory on the target machine.
   A Flash drive can be mounted as follows:

   ```
   mkdir /mnt/sdxx
   mount /dev/sdxx /mnt/sdxx
   cd $ICP_ROOT
   cp /mnt/sdxx/Telephony.L.1.0.xx_Asterisk_002.tar.gz .
   ```

3. Unpack the file using the following command:

   ```
   tar -xzvf Telephony.L.1.0.xx_Asterisk_002.tar.gz
   ```

The following output is displayed:

```
LICENCE.GPL
chan_hss.c
hss.conf
install.sh
```

4. Install the software by running the install.sh script using the following command:

   ```
   ./install.sh
   ```

   Please note that when running the install.sh script, you are asked to enter three options - 1) Asterisk installation directory, 2) Asterisk Configuration files location and 3) Echo cancellation enable or disable.

   1) For the installation directory, provide the path you have chosen for Asterisk as the <asterisk-path> value.
   2) For the configuration files location, press Enter to accept the default value, /etc/asterisk.
   3) For the Echo Cancellation option, choose YES to enable echo cancellation.

11.5.5 Making A Call

This section describes the steps required to make a call:

1. Run the scripts to load the IP Telephony drivers (refer to Section 4.5, “Installation of Build Output” on page 35).
2. Please be sure to replace the default hss.conf file in the /etc/asterisk directory with the hss.conf file, the contents of which is shown in Section 11.5.5.1. In addition, to make calls between two phones, please use the extensions.conf file, the contents of which is shown in Section 11.5.5.1, and place it in the /etc/asterisk directory. If echo cancellation was enabled during installation, turn it on or off for the configured lines.
3. Now, make Asterisk by executing only the a) make and b) make install commands (shown below) as specified in the Asterisk 1.4.13 readme file that was downloaded in Section 11.5.3, “Building Asterisk Instructions” on page 64. Executing only these two commands preserves the new configuration files and builds the HSS Channel driver.

   Execute the commands by typing:

   ```
   make
   make install
   ```

4. Run Asterisk by executing the command as specified by Asterisk 1.4.13 readme file that was downloaded in Section 11.5.3 and make sure that the HSS Channel Driver is loaded by typing:

   ```
   module show like chan_hss
   ```

   If an analog voice mezzanine card is configured, the green LEDs light up after Asterisk is launched.

   Assuming you have plugged POTS phones into HSS ports 1 and 2 on the analog voice mezzanine card, you can make a call between the phones. See Section 11.5.5.1 for example configuration file information and Section 11.5.5.2 for instructions.

11.5.5.1 Example Configuration Files

extensions.conf

```
; extensions.conf - the Asterisk dial plan
;
; Static extension configuration file, used by
; the pbx_config module. This is where you configure all your
; inbound and outbound calls in Asterisk.
;
; This configuration file is reloaded
; - With the "dialplan reload" command in the CLI
; - With the "reload" command (that reloads everything) in the CLI

;
; The "General" category is for certain variables.
;
[general]
;
; If static is set to no, or omitted, then the pbx_config will rewrite
; this file when extensions are modified.  Remember that all comments
; made in the file will be lost when that happens.
;
; XXX Not yet implemented XXX
;
static=yes
;
; if static=yes and writeprotect=no, you can save dialplan by
; CLI command "dialplan save" too
;
writeprotect=no
;
; If autofallthrough is set, then if an extension runs out of
; things to do, it will terminate the call with BUSY, CONGESTION
; or HANGUP depending on Asterisk's best guess. This is the default.
;
; If autofallthrough is not set, then if an extension runs out of
; things to do, Asterisk will wait for a new extension to be dialed
; (this is the original behavior of Asterisk 1.0 and earlier).
;
;autofallthrough=no
;
; If clearglobalvars is set, global variables will be cleared
; and reparsed on an extensions reload, or Asterisk reload.
;
; If clearglobalvars is not set, then global variables will persist
; through reloads, and even if deleted from the extensions.conf or
; one of its included files, will remain set to the previous value.
;
; NOTE: A complication sets in, if you put your global variables into
; the AEL file, instead of the extensions.conf file. With clearglobalvars
; set, a "reload" will often leave the globals vars cleared, because it
; is not unusual to have extensions.conf (which will have no globals)
; load after the extensions.ael file (where the global vars are stored).
; So, with "reload" in this particular situation, first the AEL file will
; clear and then set all the global vars, then, later, when the extensions.conf
; file is loaded, the global vars are all cleared, and then not set, because
; they are not stored in the extensions.conf file.
;
clearglobalvars=no
;
; If priorityjumping is set to 'yes', then applications that support
; 'jumping' to a different priority based on the result of their operations
; will do so (this is backwards compatible behavior with pre-1.2 releases
; of Asterisk). Individual applications can also be requested to do this
; by passing a 'j' option in their arguments.
```

```
;
;priorityjumping=yes
;
; User context is where entries from users.conf are registered.  The
; default value is 'default'
;
;userscontext=default
;
; You can include other config files, use the #include command
; (without the ';'). Note that this is different from the "include" command
; that includes contexts within other contexts. The #include command works
; in all asterisk configuration files.
;#include "filename.conf"

[default]
exten => 111,1,Dial(HSS/1)
exten => 222,1,Dial(HSS/2)
exten => 333,1,Dial(HSS/3)
exten => 444,1,Dial(HSS/4)
exten => 1234,1,Dial(SIP/10.127.144.155)
```

hss.conf

```
; HSS channel driver configuration
; This section is for HSS ports configuration
[configuration]
geo=us                  ; geo zone, eu or us. default: us
mode=SLIC               ; Mode: SLIC
port => 0
; This section is for the lines configuration
[channels]
echocancel=yes        ; Enable/disable echo cancellation. default: no
callwaiting=no        ; Enable/disable call waiting. default: no
silencesuppression=no ; Enable/disable silence suppression. default: no
context=default               ;  Incoming context
format=ulaw                   ;  Codec supported, ulaw, alaw or linear
callerid= HSS-phone1 <111>    ;  Callerid: name <extension>
line => 1                     ;  Line IDs. Can be numbers separated by
                              ;  commas, or a range separated by '-'
context=default         ;
format=alaw
callerid= HSS-phone2 <222>
line => 2
context=default         ;
format=alaw
callerid= HSS-phone3 <333>
line => 3
context=default         ;
format=alaw
callerid= HSS-phone4 <444>
line => 4
```

11.5.5.2 Dialing Between Two POTS Phones

Proceed as follows:

1. Plug in POTS phones on HSS ports 1 and 2 on analog voice mezzanine card.
2. Dial 222 on the first POTS phone to call the second POTS phone.
3. Dial 111 on the second POTS phone to call the first POTS phone.

Note: For more information on the Asterisk Dial command, refer to the Asterisk wiki page at: http://www.voip-info.org/wiki/index.php?page=Asterisk+cmd+Dial

4. To shutdown Asterisk and properly release the IP Telephony drivers, use the following command:

```
stop gracefully
```

12.0 Troubleshooting

Refer to the Release Notes for your software package for a list of known errata, implications, and workarounds.

12.1 Using a Graphics Card

Some LCD monitors do not function properly with the development board. In one case, an “Input Not Supported” message was reported upon boot completion. Try an alternative LCD monitor if video is not displayed.

12.2 Using the Serial Port

The user can bring up the system without using the Graphics card. In this case, the IO takes place over the serial port. The serial port is configured as follows:

1. Enable communication over all serial ports in the BIOS setup. Configure ports to 115200, No Parity, 8 bit, 1 stop bit and No Flow Control.
2. Use a female-female 9-pin RS232 cable to connect the CRB to a terminal emulator application such as HyperTerminal* in Windows. Configure the terminal emulator to 115200, No Parity, 8 bit, 1 stop bit and No Flow Control.
3. Edit grub.conf (the bootloader, enables Grub output on the serial port and console logging via the serial port):
   a. Make a backup of /etc/grub.conf before doing any changes.
   b. Edit that file and just underneath "timeout=5" add:

      ```
      serial --unit=0 --speed=115200 --word=8 --parity=no --stop=1
      terminal --timeout=0 serial console
      ```

      At the end of each “kernel” line append:

      ```
      console=tty0 console=ttyS0,115200
      ```

   c. If the user does not want a graphic card, they can boot without one, provided they modify the grub.conf file and comment out the line:

      ```
      splashimage=(hd0,0)/grub/splash.xpm.gz
      ```

      Otherwise, if no graphics card is available, the system will hang at boot.
4. Edit /etc/inittab (enables shell login from the serial port):
   a. Make a backup of /etc/inittab before making any changes and edit the file.

b. Just below the line “Run gettys in standard runlevels”, add the following line:

```
so:2345:respawn:/sbin/agetty -L -f /etc/issueserial 115200 ttyS0 v 115200
vt100-nav
```

5. You may have to edit /etc/securetty and add the lines:

```
ttyS0
ttyS1
```

6. Remove the graphics card and restart the system.

13.0 Glossary

AHCI	Advanced Host Controller Interface
ASD	Acceleration System Driver
ASU	Acceleration Services Unit
CAN	Controller Area Network
CDRAM	Coherent DRAM
DIMM	Dual Inline Memory Module
EDMA	Enhanced Direct Memory Access
EFI	Extensible Firmware Interface
FWH	Firmware Hub
GbE	Gigabit Ethernet
GCU	Global Configuration Unit
GPIO	General Purpose Input Output
HDLC	High-Level Data Link Control
IEEE	Institute of Electrical and Electronics Engineers
NCDRAM	Non-Coherent DRAM
OCF	OpenBSD Cryptographic Framework
OCF Patch	Patch file used to include OCF files during kernel rebuild
OCF Shim	A kernel module that allows the OCF module to offload cryptographic functions to EP80579 cryptographic accelerators.
POST	Power On Self Test
RNG	Random Number Generator
SATA	Serial Advanced Technology Attachment
SMbus	System Management Bus
TDM	Time Division Multiplexing
WDT	Watchdog Timer

Appendix A Build and Install Openswan

This chapter provides instructions for building and installing Openswan on the development board. The next chapter describes how the EP80579 security software thus configured can be used to set up a sample VPN application.

Note: It is not required to install Openswan to use the Intel® EP80579 Software for Security Applications on Intel® QuickAssist Technology Cryptographic API.

A.1 Environment Setup

Define the following environment variables:

```
export ICP_ROOT=/EP805XX_release
export KERNEL_SOURCE_ROOT=/usr/src/redhat/BUILD/kernel-2.6.18/linux-2.6.18.i386
```

Note: As per the instruction in the previous chapters, the kernel source are untarred at KERNEL_SOURCE_ROOT and other sources are untarred at ICP_ROOT.

A.2 Building and Installing the GMP Library

Openswan uses a large number and arbitrary arithmetic precision library called GMP for certain mathematical functions.

1. Download the GMP library from the following location:
   http://ftp.sunet.se/pub/gnu/gmp/gmp-4.2.1.tar.gz

2. Copy the file to the $ICP_ROOT directory and extract the downloaded GMP package:

   ```
   cd $ICP_ROOT
   tar -xvzf gmp-4.2.1.tar.gz
   ```

3. Change directory to the newly created GMP directory.

   ```
   cd gmp-4.2.1
   ```

4. Run the autoconfigure script.

   ```
   ./configure
   ```

5. Compile the code.

   ```
   make
   ```

6. Run the required tests.

   ```
   make check
   ```

7. If successful, install the package.

```
make install
```

8. Add the following configuration to recognize libraries in /usr/local/lib:

```
echo "/usr/local/lib" > /etc/ld.so.conf.d/gmp-4.2.1.conf
```

9. Update the shared libraries.

```
ldconfig
```

10. Execute the following command to verify that the gmp library is recognized.

```
ldconfig -p | grep gmp
```

A.3 Building and Installing OpenSSL

Note: When Openswan is patched to work with the OCF, it requires the "bignum" library from OpenSSL. OpenSSL is installed simply to access the "bignum" library that comes with it. OpenSSL cryptography features can be accelerated using QuickAssist technology by using the "-engine cryptodev" option.

1. Download the OpenSSL package from the following location:

   http://www.openssl.org/source/openssl-0.9.8g.tar.gz

Note: Note that the downloaded package has the extension tar.tar and not tar.gz.

2. Download the OCF patch for Linux from the following location:

   http://heanet.dl.sourceforge.net/sourceforge/ocf-linux/ocf-linux-20080917.tar.gz

3. Extract the OCF patch for openssl-0.9.8g from the above tarball using the following command:

```
tar -xvzf ocf-linux-20080917.tar.gz ocf-linux-20080917/openssl-0.9.8g.patch
```

Note: Only the patch file for OpenSSL is extracted from this tarball. Rest of the document still refers to OCF patches for the release OCF 20070727

4. Copy the file to the $ICP_ROOT directory and extract the downloaded OpenSSL package. Execute the following commands:

```
cd $ICP_ROOT
tar -xvf openssl-0.9.8g.tar.tar
cd openssl-0.9.8g
```

5. Apply the patches using the following commands:

```
patch -p1 < ../ocf-linux-20080917/openssl-0.9.8g.patch
patch -p1 < $ICP_ROOT/OpenSourcePatches/ocf-openssl-
0.9.8g_DSA_verify_fix.patch
```

6. Run the "autoconfigure" script with some additional parameters related to the shared library. Note the file name is Configure with a capital C.

```
./Configure threads shared --with-cryptodev --with-cryptodev-digests --
prefix=/usr/local/ssl_0.9.8g linux-elf -g
```

7. Compile and install the package:

```
make
make install
```

8. To ensure that the libraries are correctly installed, copy the libcrypto.so and libssl.so files to the /usr/lib directory and add the libraries into ldconfig's path:

```
echo "/usr/local/ssl_0.9.8g/lib" >> /etc/ld.so.conf.d/qt-i386.conf
ldconfig
```

9. Execute the following commands to ensure that the openssl 09.8g library is recognized

```
ldconfig -p | grep libssl
ldconfig -p | grep libcrypto
```

10. Execute the following command to ensure that openssl binary is using the correct libraries:

```
ldd /usr/local/ssl_0.9.8g/bin/openssl
```

The output should look like the following:

```
libssl.so.0.9.8 => /usr/local/ssl_0.9.8e/lib/libssl.so.0.9.8 (.......)
libcrypto.so.0.9.8 => /usr/local/ssl_0.9.8e/lib/libcrypto.so.0.9.8
(.......)
```

11. Execute the following commands so that the newly installed openssl is properly recognized by other components:

```
cd /usr/local
ln -f -s ssl_0.9.8g openssl
ln -f -s ssl_0.9.8g ssl
export PATH=/usr/local/ssl_0.9.8g/bin:${PATH}
export LD_LIBRARY_PATH=/usr/local/ssl_0.9.8g/lib:${LD_LIBRARY_PATH}
```

12. To ensure the OpenSSL headers are correctly installed, copy the contents of the ./include/openssl/ directory into the /usr/include/openssl directory

```
cd $ICP_ROOT/openssl-0.9.8g
mkdir -p /usr/include/openssl
cp -L ./include/openssl/* /usr/include/openssl
```

Note: Overwrite existing files if required.

13. Verify that the correct version of OpenSSL has been installed using the following command:

```
openssl version
```

The output should look like the following:

```
OpenSSL 0.9.8g 23 Oct 2008
```

14. Install OCF stack as described in Chapter 5.0, "Runtime Configuration". Verify the OCF access from OpenSSL engine using the following command:

```
openssl engine -t
```

The output should look like the following:

```
(cryptodev) BSD cryptodev engine [available]
```

15. Check the crypto operations supported by the OCF engine using the following command:

```
openssl engine -c
```

The output should look like the following:

```
(cryptodev) BSD cryptodev engine [RSA, DSA, DH, DES-CBC, DES-EDE3-CBC, AES-128-CBC, AES-256-CBC, hmacWithSHA1, MD5, SHA1]
```

16. Verify that an OpenSSL speed test can be run by executing the following command:

```
openssl speed -evp des3 -elapsed -engine cryptodev
```

Note: Note that loading the module cryptosoft.ko only will result in OCF using crypto software library. Whereas, loading the module icp_ocf.ko will result in OCF using the hardware accelerator.

A.4 Building and Installing Openswan

A.4.1 Prepare to Build Openswan

1. Pluto requires the flags: HAVE_OCF, HAVE_OPENSSL to be enabled to use OCF for crypto functionality

```
export KERNELSRC=$KERNEL_SOURCE_ROOT
cd $KERNELSRC
cp ./include/linux/autoconf.h ./include/linux/autoconf.h.sav
echo "#define CONFIG_KLIPS_OCF 1" >> ./include/linux/autoconf.h
make modules_prepare
```

2. Copy the updated file into the system include directory as follows:

```
cp ./include/linux/autoconf.h /usr/include/linux/autoconf.h
```

Note: You may need to create some of the referenced directories first.

A.4.2 Building and Installing Openswan

1. Download the Openswan package and OCF-related patch from the following locations:
   http://www.openswan.org/download/old/openswan-2.4.9.tar.gz
   http://heanet.dl.sourceforge.net/sourceforge/ocf-linux/ocf-openswan-2.4.9-20070727.patch.gz
2. Copy the files to the $ICP_ROOT directory and extract the downloaded Openswan package. Execute the following commands:

```
cd $ICP_ROOT
tar -xvzf openswan-2.4.9.tar.gz
gunzip ocf-openswan-2.4.9-20070727.patch.gz
```

3. Change directory to the newly created openswan-2.4.9 directory and apply the patch to enable OCF usage.

```
cd openswan-2.4.9
patch -p0 < ../ocf-openswan-2.4.9-20070727.patch
```

4. Again, to backport features and bug fixes, apply the following patch:

```
patch -p1 < $ICP_ROOT/OpenSourcePatches/ocf-openswan-2.4.9-20070727-session-migration-backport.patch
```

5. Disable the batch mode in OCF by editing the file ./linux/net/ipsec/ipsec_ocf.c. Set the constant USE_BATCH to zero. For example,

```
#define USE_BATCH 0
```

6. Build Openswan.

Note: The $KERNELSRC variable must be set as described in Section A.4.1 before the build will work.

Note: The make minstall command installs KLIPS permanently and avoids a change between KLIPS and the Netkey stack.

```
make clean
make CONFIG_KLIPS_OCF=y DECLARE_TASKLET=y module >build_klips.log 2>&1
make minstall
make HAVE_OCF_AND_OPENSSL=y programs >build_pluto.log 2>&1
make HAVE_OCF_AND_OPENSSL=y install
```

This first compiles and installs the kernel module, then compiles and installs the userland component.

7. Update any new libraries and update the kernel module dependencies.

```
ldconfig
depmod -a
```

8. Openswan installs the S47ipsec startup file in the /etc/rc3.d directory. To prevent it auto-starting on every reboot, execute the following command:

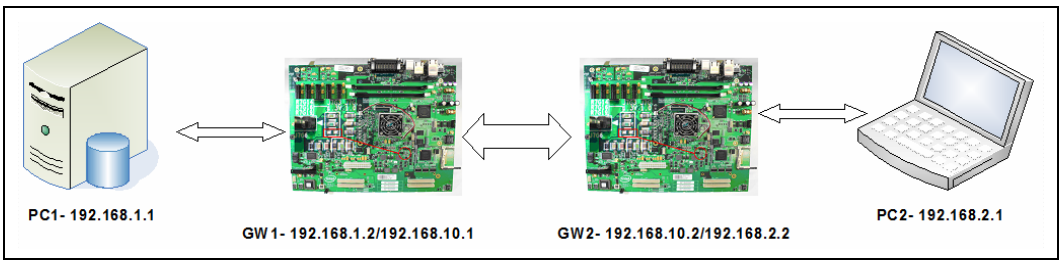
```
chkconfig ipsec off
```

Appendix B Configuring Sample VPN Application

This chapter describes how to accelerate an Openswan IPSec tunnel between an EP80579 gateway GW1 and another IA gateway GW2 using OCF and the OCF Shim for the EP80579 with QuickAssist. The resulting offload of the crypto function can be used to increase the Openswan* throughput or release cycles for other user applications.

The configuration is shown in Figure 9 and it assumes that the two PCs, PC1 and PC2, are running Linux*.

Figure 9. VPN Example



B.1 Configure IPSec on Both Gateways

Perform the following steps to set up an IPSec on gateways GW1 and GW2:

1. If an existing system with IPSec implementation is not available, then install one as described in Section B.4, “Installing the OS and Openswan on Gateway GW2” on page 82.
2. Install two Gigabit NICs on gateway GW2.
3. Connect the gateways GW1, GW2 and the PCs PC1 and PC2 as shown in Figure 9 using crossover CAT5 cables. Alternatively, the user could use different hubs or switches to make the connections.
4. Build and install Openswan on gateway GW2.
5. Configure the IP addresses for the EP80579 internal Gigabit ports and enable IP forwarding on GW1 using the following commands:

```
ifconfig eth0 192.168.10.1/24 up
ifconfig eth1 192.168.1.2/24 up
echo 1 > /proc/sys/net/ipv4/ip_forward
```

Note: In Linux, the Ethernet ports TT, SS and UU in Figure 4 on page 17 appears as eth0, eth1 and eth2 respectively.

6. Configure the IP addresses for the IA Gigabit ports and enable IP forwarding on gateway GW2 using the following example commands:

```
ifconfig eth0 192.168.10.2/24 up
ifconfig eth1 192.168.2.2/24 up
echo 1 > /proc/sys/net/ipv4/ip_forward
```

7. Configure networking on PC1 using the following example commands:

```
ifconfig eth0 192.168.1.1/24 up
route add default gw 192.168.1.2
```

8. Configure networking on PC2 using the following example commands:

```
ifconfig eth0 192.168.2.1/24 up
route add default gw 192.168.2.2
```

9. If the network is configured properly then the user should be able to ping each of the Ethernet ports configured so far from PC1, PC2, GW1 and GW2.
10. Start OCF on GW1 as described in Section 5.2, “Loading OCF and OCF Shim (Optional for IP Telephony Applications)” on page 38.
11. Edit the /etc/ipsec.conf file on both Gateways based on details provided in Section B.3, “Configuring Crypto Parameters in ipsec.conf” on page 79.
12. Stop any IPSec service on GW1 and verify that it has stopped:

```
service ipsec stop
service ipsec status
```

13. Restart IPSec and verify success as follows:

```
service ipsec start
service ipsec status
```

14. Stop, then restart IPSec on GW2.

B.1.1 Testing if IPSec is Configured Correctly

1. The route command on each Gateway should show an additional entry for device ipsec0. Execute the following command:

```
route
```

2. Execute the following command to display the supported ciphers:

```
cat /proc/net/pf_key_supported
```

   In the right-most column, a list of ciphers prefixed with “ocf-” (for example, “ocf-sha1hmac”) is displayed. This indicates that Openswan has successfully detected OCF and can use it for cryptographic calls. If every entry in the right-most column reads “unknown”, the installation failed.

3. Check the IPSec setup on each gateway using the following command:

```
ipsec verify
```

   This command indicates the IPSec version, if “Pluto” is running and which IPSec network stack is being used -- NETKEY or KLIPS.

Note: If the output indicates that Pluto is not running, then execute the following commands as a workaround:

```
ln -s /usr/local/lib/libgmp.so.3 /lib/libgmp.so.3
```

Note: Another troubleshooting technique to narrow down an IPSec connectivity issue is to load OCF and the associated crypto software library instead of the EP80579 OCF shim. This helps to isolate the issue to Openswan or OCF Shim and LAC software. Execute the following commands to install OCF to use crypto software library:

```
modprobe ocf
modprobe cryptodev
modprobe cryptosoft
```

B.2 Set Up a Tunnel Between Gateways

Note: Execute the following commands on every reset in order to support 1000 VPN tunnels and prevent ARP table from overflowing:

```
echo 2048 > /proc/sys/net/ipv4/neigh/default/gc_thresh1
echo 4096 > /proc/sys/net/ipv4/neigh/default/gc_thresh2
echo 8192 > /proc/sys/net/ipv4/neigh/default/gc_thresh3
```

1. The following command can be used to set up, bring down and tear down a tunnel:

   ```
   ipsec auto --up <name_of_tunnel>
   ipsec auto --down <name_of_tunnel>
   ipsec auto --delete <name_of_tunnel>
   ```

   where <name_of_tunnel> is defined in the /etc/ipsec.conf file. See Section B.3.2.1 for more information.

Note: This command to enable (up) need be executed only on one Gateway. However, the command to disable (down) and tear down the tunnel should be executed on both Gateways.

2. If the tunnel has been setup, the route command should show two entries with the interface set to device ipsec0.
3. Execute the following commands to trace the packet flow across the tunnel:

   ```
   tcpdump -i eth<x>
   tcpdump -i ipsec0
   ```

   If PC1 pings PC2 across the tunnel, a packet trace of eth<x> should show ESP packets being sent across and ipsec0 should show ICMP messges being sent.
4. The number of packets sent across the tunnel can be counted using the following commands. The second command updates the count every <x> seconds.

   ```
   ipsec eroute
   watch -n <x secs> ipsec eroute
   ```

Using a web search engine, search for “openswan ipsec.conf” for details on each field in the file.

B.3 Configuring Crypto Parameters in ipsec.conf

Openswan VPN application uses two configuration files: The file ipsec.secrets contains the secret keys used for authentication and the file ipsec.conf contains all other configuration information for the tunnel setup.

The “authby” parameter in the ipsec.conf file provides two ways for the IKE on each host to authenticate each other:

- “rsasig”, which uses the RSA algorithm
- “secret”, which uses a Pre-Shared Key (PSK)

RSA is a little more difficult to setup, but it is the preferred because it is the more secure. PSK is slightly easier since it uses a shared secret as a key with the associated security risks and manual key exchange issues.

The next two sections describe how to create and incorporate secret keys into the configuration files and the last section provides an usable configuration files that uses PSK authentication.

B.3.1 RSA SIG

To use RSA, execute the following on each of gateway GW1 and GW2:

1. Set “authby=rsasig” in the ipsec.conf file.
2. Run the following command to generate the RSA key:

   ```
   ipsec rsasigkey --verbose 2192 > rsakey.tmp
   ```

   The following is a sample output:

   ```
   # 1024 bits, Fri Feb  4 05:05:19 2000
   # for signatures only, UNSAFE FOR ENCRYPTION
   #pubkey=0x010395daee1be05f3038ae529ef2668afd79f5ff1b16203c9ceeaef801c..
   Modulus:0x95daee1be05f3038ae529ef2668afd79f5feaef801cecfb51a6ecc08890..
   PublicExponent: 0x03
   # everything after this point is secret
   PrivateExponent:0x63e74967eaea2025c98c69f6ef0753a6a3ff676415771324dd3..
   Prime1:0xc5b471a88b025dd09d4bd7b61840f9c5d45546bea3ccc7b744254f6f0b84..
   Prime2:0xc20a99feeafe79763f2f45b0e96cb4aef8918ca333a326d3f6dc2c72b753..
   Exponent1:0x83cda11b0756e93e1674fff4512e8d8e2f29c2888524d818df9f5d02f..
   Exponent2:0x815c66a9f1fefba44b6c2b1246f5061086ccd176f37f9e81da1cf8ceb..
   Coefficient:0x10d954c9e2b8d11f4db1b233ef37ff0a3c5d30186520f1753a7e325..
   ```

3. Copy the contents of rsakey.tmp to /etc/ipsec.secrets:

   ```
   cp -f rsakey.tmp /etc/ipsec.secrets
   ```

4. Without deleting any contents, edit the /etc/ipsec.secrets file to insert a new line at the beginning and at the end.

   The edited file should look like the following:

```
: RSA {
   # 1024 bits, Fri Feb  4 05:05:19 2000
   # for signatures only, UNSAFE FOR ENCRYPTION
   #pubkey=0x010395daee1be05f3038ae529ef2668afd79f5ff1b16203c9ceeaef801c..
   Modulus:0x95daee1be05f3038ae529ef2668afd79f5feaef801cecfb51a6ecc08890..
   PublicExponent: 0x03
   # everything after this point is secret
   PrivateExponent:0x63e74967eaea2025c98c69f6ef0753a6a3ff676415771324dd3..
   Prime1:0xc5b471a88b025dd09d4bd7b61840f9c5d45546bea3ccc7b744254f6f0b84..
   Prime2:0xc20a99feeafe79763f2f45b0e96cb4aef8918ca333a326d3f6dc2c72b753..
   Exponent1:0x83cda11b0756e93e1674fff4512e8d8e2f29c2888524d818df9f5d02f..
   Exponent2:0x815c66a9f1fefba44b6c2b1246f5061086ccd176f37f9e81da1cf8ceb..
   Coefficient:0x10d954c9e2b8d11f4db1b233ef37ff0a3c5d30186520f1753a7e325..
   }
```

Note: Line 1: ‘:’ is on column 1; space between ‘:’ and ‘RSA’ ; space between ‘RSA’ and ‘{‘

Last Line: ‘}’ should NOT be in column 1.

Note: The comments lines after ‘#’ are important. For example, the public key is stored as a comment.

5. The public key information from /etc/ipsec.secrets on gateway GW1 should be copied to the field ‘leftrsasigkey’ in /etc/ipsec.conf on both GW1 and GW2. Similarly, the public key information from /etc/ipsec.secrets on gateway GW2

should be copied to the field ‘rightrsasigkey’ in /etc/ipsec.conf on both GW1 and GW2.

Since the public key is a large string and manual copy is error prone, use the following command to extract the exact string:

Run the following command on GW1 to extract the public key:

```
ipsec showhostkey --left
```

Run the following command on GW2 to extract the public key:

```
ipsec showhostkey --right
```

After the edits, the file /etc/ipsec.conf on both gateways GW1 and GW2 should be identical and the relevant fields should look like the following:

```
.............
conn gw-to-gw
left=192.168.10.1
leftsubnet=192.168.1.0/24
leftrsasigkey=0x010395daee1be05f3038ae529ef2668afd79f5ff1b16203c9....
right=192.168.10.2
rightsubnet=192.168.2.0/24
rightrsasigkey=0x01037631b81f00d5e6f888c542d44dbb784cd3646f084ee....
esp=3des-sha1-96
authby=rsasig
type=tunnel
.............
```

Note: When entering the left and right rsasigkeys into the ipsec.conf file be careful not to include any spaces after the key, as this will result in error when trying to start the ipsec service.

B.3.2 PSK

Note: Sample PSK secret files and corresponding config files are available in the software package. See section Section B.3.2.1 for details.

To use PSK, execute the following steps:

1. Set the following in /etc/ipsec.conf on both gateways:

   ```
   authby=secret
   ```

2. Generate a secret on any one of the gateways by executing:

   ```
   ipsec ranbits --continuous 128 > somerandommonkey.tmp
   ```

3. Add the new secret to “/etc/ipsec.secrets” on gateway GW1. The resulting file should look like:

   ```
   192.168.10.1 192.168.10.2 : PSK 0xb2463f384577f95448052aad46496b7ec
   ```

4. Add the new secret to “/etc/ipsec.secrets” on gateway GW2. The resulting file should look like:

   ```
   192.168.10.2 192.168.10.1 : PSK 0xb2463f384577f95448052aad46496b7ec
   ```

Note: The PSK secret key should be the same on both gateways, but the IP address pairs should be reversed

B.3.2.1 Sample Usable Configuration and Secret Files

Usable ipsec.conf and ipsec.secrets file can be found in $ICP_ROOT/OpenswanConfigFiles.

Execute the following command to copy the files to GW1:

```
cp $ICP_ROOT/OpenswanConfigFiles/ipsec.conf /etc/

cp $ICP_ROOT/OpenswanConfigFiles/ipsec.left.secrets /etc/ipsec.secrets
```

Execute the following command to copy the files to GW2:

```
cp $ICP_ROOT/OpenswanConfigFiles/ipsec.conf /etc/

cp $ICP_ROOT/OpenswanConfigFiles/ipsec.right.secrets /etc/ipsec.secrets
```

B.4 Installing the OS and Openswan on Gateway GW2

If an existing system where an IPSec implementation is not available, then perform the following steps to install a RHEL 5.0 and Openswan on top of it:

1. Select a system that satisfies the basic hardware requirements to install a RHEL 5.0 system.
2. Follow the instructions in Chapter 3.0, “Installing the OS on a Development Board”, but do not install any patches.
3. Install the gmp library by following the instructions in Section A.1, “Environment Setup” on page 72 and Section A.4, “Building and Installing Openswan” on page 75.
4. Download the Openswan package from the following location:
   http://www.openswan.org/download/old/openswan-2.4.9.tar.gz

   to $ICP_ROOT and extract it using the following commands:

   ```
   cd $ICP_ROOT
   tar -xvzf openswan-2.4.9.tar.gz
   ```

5. Build Openswan using the following commands:

   ```
   cd openswan-2.4.9
   make module minstall programs install
   cd $ICP_ROOT
   ldconfig
   depmod -a
   ```

§ §